Data structures - pascal

Safiya Najeh Noori Al-Badri

Bagdad-Iraq

بسم الله الرحمن الرحيم

هياكل البيانات – باسكال

Data Structures – Pascal

تأليف وإعداد وتقديم:

صفية ناجح نوري البدري

المقدمة

تستخدم أنظمة الحواسيب بصورة أساسية في معالجة البيانات وعليه فإن طريقة تنظيم هذه البيانات تؤثر على سرعة المعالجة حيث أن سرعة المعالجة للبيانات والحصول على النتائج يعتبر من المتطلبات الأساسية في أنظمة الحواسيب ، وأيضاً تنظم هذه البيانات يؤثر على مدى استقلال الذاكرة الرئيسية في أنظمة الحاسوب يجب أن يكون مرتفع إلى حد ما .

مقدمة إلى تراكيب البيانات:–

المعلومات والبيانات :-

تعرف البيانات على أنها مجموعة من الحقائق والأفكار التي لم يتم معالجتها وتعرف بأنها المادة الخام للمعلومات ، وحتى تصبح البيانات معلومات لابد أن تعالجهذه البيانات بطريقة معينة لتعطي بيانات ذات فائدة معينة تسمي بالمعلومات ، تمتاز المعلومات عن البيانات في أنها مرتبة ، مصنفه ملخص وذات فائدة معينة .

من أهم العمليات التي يمكن استخدامها لمعالجة البيانات نذكر مايلي :-

١- تجميع البيانات من مصادرها المختلفة .
٢- التحقق من صحة البيانات .
٣- ترميز البيانات والتأكد من صحة الترميز .
٤- تخزين البيانات على وسائط تخزين تمتاز بالسعة العالية وبسهولة استرحام البيانات منها وسرعة الوصول إليها .
٥- ترتيب وتنظيم البيانات في هياكل بيانات محددة لتسهل عملية فهم ودراسة البيانات .
٦- العمليات الحسابية والمنطقية التي تتم على البيانات باستخدام العلاقات المختلفة .
٧- فرز أو دمج البيانات معاً .
٨- كتابة التقارير المختلفة من البيانات التي تمت معالجتها .

وأكثر ما يهمنا هنا تنظيم البيانات في هياكل بيانات محددة . فماذا نقصد بهياكل البيانات ؟ ما هي أنواع هياكل كل البيانات ؟

الهيكلية خاصية من خصائص البيانات :-

دائماً ما نجد أن الأشياء تمتاز بمجموعة صفات ، وهذه الصفات تعتبر الخصائص المميزة والمحددة للأشياء ، وتنتظم هذه الصفات طبيعياً بشكل بنائي منظم .

فمثلاً أن أسماء أفراد العائلة تنتظم تحت بناء يشبه الشجرة "tree" . حيث يعبر الجد عن الجذر فيها والأبناء عن الفروع وكذلك الأحفاد وهكذا .

وكمثال آخر ينتظم الناس في طابور لشراء حاجة ما يعبر الواقف أولاً عن رأس الطابور والأخير عن ذيل الطابور ودليل التلفون وغيرها من الأمثله .

كل هذه الأمثلة تبين لنا حقيقة واحدة ، هي أن المعلومات تمتاز بالترتيب هذا الترتيب يعرف بالهيكل أو هياكل البيانات "data structure" .

ومن هنا نلخص الى تعريف هيكل البيانات ، حيث أنه تشكيل منظم لمجموعة من البيانات التي تشترك بصفة أو أكثر فيما بينها وذلك لتؤدي غرضاً محدداً حول شيء أو مجموعة أشياء محددة .

ويمكن أن نجد هذا التشكيل المنظم في الأمثلة التي سقناها سابقاً فمثلاً ، أن دليل التلفون يضم في كل صفحة من صفحاته عمودين أحدهما يدل على أسماء أشخاص معينين ، أو مؤسسات أو هيئات محددة والعود الأخر يدل على أرقام الهواتف التي يملكها الأفراد أو المؤسسات أو الهيئات الموجودة في المعود الأول .

ويكتمل هذا التشكيل المنظم بترتيب العمود الأول أبجدياً وذلك لتسهيل عملية البحث عن فرد أو مؤسسة أو هيئة معينة .

إن هيكل البيانات ضروري لمعالجة البيانات نظراً لما تتمتع به هياكل البيانات من مميزات تساعد في الإسراع في الوصول إلى البيانات ومعالجتها ، فمثلاً استخدام المؤشرات في هياكل البيانات يساعد في عملية الحذف والإضافة ، فحذف عنصر معين من الهيكل لا يؤدي إلى إعادة ترتيب العناصر الأخرى ويكفي إجراء عملية التعديل على المؤشر وكذلك الحال بالنسبة للإضافة ، وبمعني آخر يمكن حفظ عناصر البيانات للهيكل الواحد في أماكن مختلفة مكتفين فقط بإضافة مؤشر للعنصر بحيث يشير هذا المؤشر إلى موقع العنصر التابع في الهيكل مما يؤدي إلى الإسراع في تنفيذ كافة عمليات المعالجة لهذه العناصر ، وسوف نتناول لاحقاً هذه الأشياء بإسهاب .

كما أسلفنا فان عملية تجميع البيانات في هياكل مختلفة تؤدي إلى الإسراع في عمليات المعالجة فإن طريقة التخزين تلعب هي الأخرى دوراً هاماً في زيادة سرعة المعالجة لهذا لا بد من الإشارة إلى التركيب الفيزيائي والتركيب المنطقي للبيانات .

فالمعالجة البيانات فانه يتطلب وجود برنامج خاص بالمعالجة يجب أن يكون موجود في الذاكرة الرئيسية أما البيانات فيمكن أن توجد على وسط تخزين خارجي " وحدة إدخال " يتم نقلها جزءاً إلى الذاكرة الرئيسية بقراءتها ومن ثم معالجتها .

وكما هو معروف فإن الذاكرة الرئيسية تنقسم إلى خلايا معنوية وتعتمد سرعة الوصول إلى المعلومات على طريقة العنونة المستخدمة ؛ أما بالنسبة للعامل الثاني فإن سرعة الوصول إلى

المعلومات لإحضارها فتعتمد على سرعة الوسط المخزنة عليه ، أما ما يهمنا هنا هو التقليل من وقت تبادل المعلومات بين الذاكرة الرئيسية ووحدات الإدخال والإخراج ولهذا لابد من الإشارة إلى التركيب الفيزيائي والمنطقي للبيانات ، حيث يبين التركيب الفيزيائي الشكل المخزنة به البيانات على وحدات التخزين الثانوية ويسمي هذا الشكل بالوحدة الفيزيائية أما وحدة البيانات " الهيكل مثلاً " فتسمي بالوحدة المنطقية ، وسوف نوضح هذا من خلال المثال التالي :

لنفرض أن ملف الطلاب يتكون من ١٠٠٠ سجل وأن معدل التكتل يساوي ٥ (عدد السجلات في الكتلة الواحدة = ٥ سجلات) على فرض أن الملف مخزن على قرص مغنطيسي . لهذا ينظر إلى الكتل " blocks " على أنها وحدات فيزيائية أما السجل داخل الكتل " block " فينظر إليه كوحدة منطقية .

واستخدام الوحدات الفيزيائية في المعالجة يؤدي إلى زيادة السرعة في المعالجة وذلك بالتقليل من عمليات القراءة فبدلاً من قراءة خمسة سجلات " وحدات منطقية " يتم قراءة وحدة فيزيائية واحدة حيث تنتقل هذه الوحدة الفيزيائية مرة واحدة إلى الذاكرة الرئيسية وتخزن هناك في مناطق تسمي بمناطق التخزين المؤقتة " buffer " حيث يتم نقل وتبادل المعلومات إلى داخل الذاكرة الرئيسية وعملية التبادل هنا تأخذ وقت أقل .

ولزيادة سرعة المعالجة يمكن تنظيم طابور " queue " من مناطق التخزين المؤقتة " queue buffer " إذا كان حجم الذاكرة يسمح بذلك .

أنواع هياكل البيانات :-

تنقسم هياكل البيانات إلى نوعين محددين هما ، هياكل بيانات ثابتة " Static data structur " وهياكل متغيرة " Dynamic data structur " وذلك تبعاً لثبات حجم البيانات أو تغيرها .

أما هياكل كل البيانات الثابتة فهي تلك المجموعة من الهياكل التي تحتوي على عدد من العناصر المحددة ، بحيث لا يزيد هذا العدد بالإضافة ولا ينقص بالحذف حيث أن كلا العمليتين غير مسموح بهما وتمتاز بعيوب عدة أهمها :-

- ضرورة توفير العدد اللازم والمتتابع من مواقع الذاكرة لتخزين عناصر الهيكل .
- عدد المواقع المخصصة للهيكل ثابتة ولا تتغير .
- سرعة المعالجة بطيئة وخاصة سرعة تنفيذ عمليات الإضافة والحذف والتي تعتبر من أكثر العمليات المنفذه على هيكل البيانات حيث يصاحب تنفيذ هذه العمليات تنفيذ عمليات التحريك اللازمة للبيانات .

بينما نجد في هياكل البيانات المتغيرة تغييراً في حجمها وذلك لإمكانية الإضافة والحذف فيها ، مع تطور أنظمة الحاسوب أصبح بالإمكان معالجة عناصر الهيكل باستخدام المؤشرات " Pointers " وبالتالي ظهرت مفاهيم هياكل البيانات الديناميكية ، إن اسخدام هذه الهياكل لتنظيم البيانات أدي إلى التخلص من الكثير من العيوب المصاحبة لإستخدام الهياكل الثابتة وتوفر الهياكل الديناميكية :-

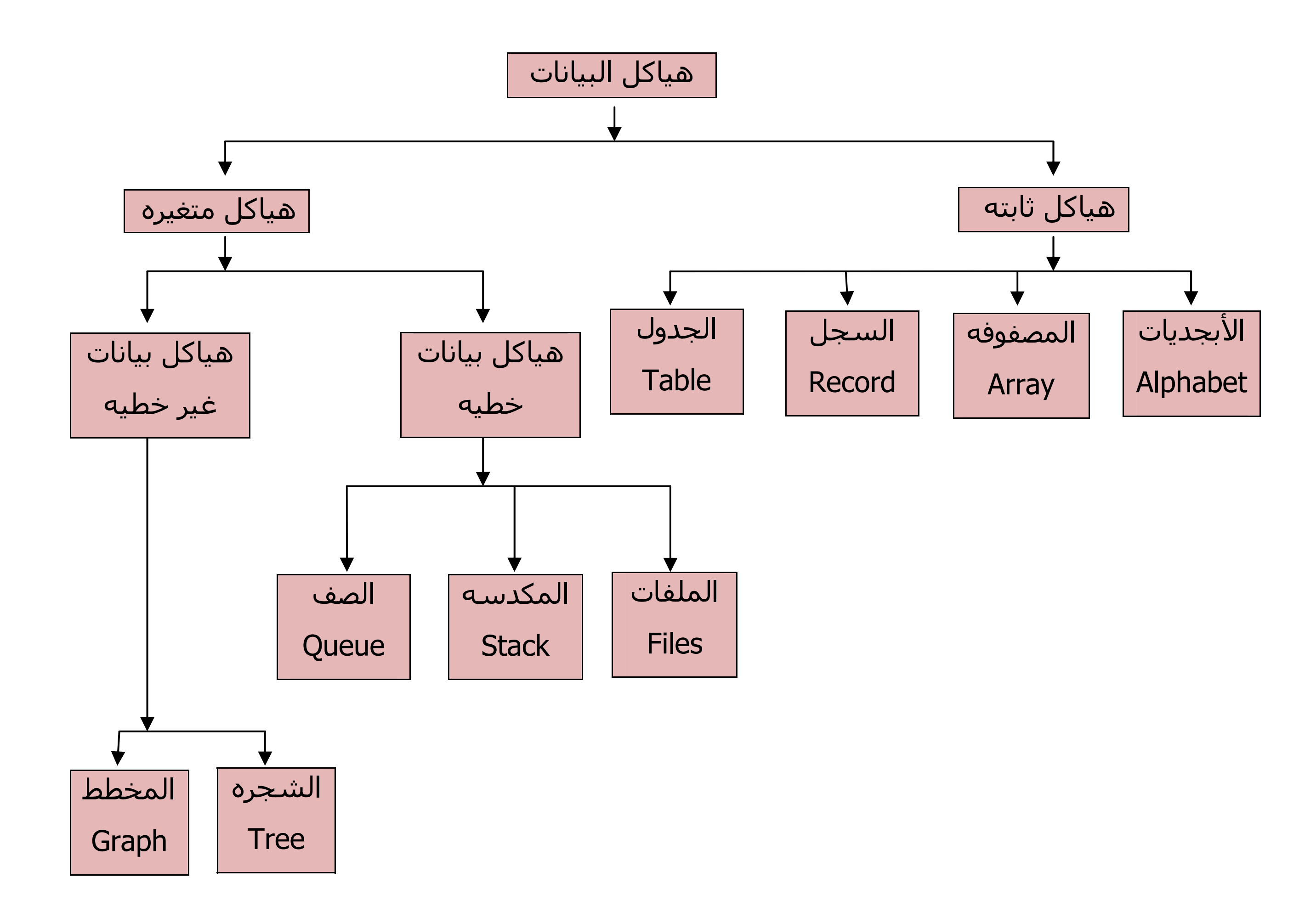
- سرعة معالجة عالية .
- استغلال الذاكرة بشكل أفضل إذ ليس بالضرورة توفير مواقع متتابعة في الذاكرة الرئيسية لتخزين عناصر الهيكل .

- حجم الهيكل الديناميكي غير ثابت ويعتمد فقط على حجم الذاكرة المتوفر .

تستخدم في عملية تنظيم البيانات هياكل مختلفة " سواء كانت ثابته أو ديناميكية " ولكل هيكل من هذه الهياكل تطبيقاته المختلفة وطرق معالجه خاصة به ، وعليه سوف نطلع القارئ في هذا الكتاب على هياكل البيانات المختلفة مركزين بذلك على :-

- مفهوم الهيكل .
- خوارزميات عمليات معالجة الهيكل وتنفيذها .
- تطبيقات الهيكل .

الشكل التالي يبين تقسيمات هياكل البيانات المخلفة :-



الوحدة الأولى

السلاسل والمصفوفات

Strings & Arrays

السلاسل والمصفوفات (Strings & Arrays)

تعتبر السلاسل الرمزية والمصفوفات من الهياكل الثابتة "static" ولهذه الهياكل تطبيقات متعددة في معالجة البيانات ونظراً لأهمية هذه الهياكل فقد خصصنا هذه الوحدة لعرض مفهوم هذه الهياكل وكيفية تحويلها من هياكل ثابته "Static " الى هياكل متحركة " dynamic " .

(١-١) السلاسل String :-

السلسلة عبارة عن مجموعة من الرموز فمثلاً :

ABCDEF

1234567

+ - * / =

THES IS ASTRING

&5{ABCDEH!

عبارة عن سلاسل رمزية مختلفة ومن أهم العمليات التي يمكن تطبيقها على السلاسل مايلي :

١- التخصيص (Assgimment) سلسلة في سلسة أخرى فمثلاً STR1 := STR2 يتم نسخ محتوي السلسلة "STR2 " في "STR1 " بدون أن يتغير محتوي "STR2 " .
٢- الدمج (Concatenatin) حيث يتم دمج محتوى سلسلتين لإعطاء سلسلة جديدة ، فمثلاً دمج السلسلة "ABC " والسلسلة "DEF " لنعطي السلسلة "ABCDEF " .
٣- البحث (Patternmatching) حيث يتم البحث عن تكرار سلسلة في سلسلة أخرى ، فمثلاً السلسلة "BASE " تتكرر في السلسلة "MODERN BASE " ابتداء من الموقع ٨ .
٤- تجزئة السلسلة (Substring operains) حيث تحدد السلسة الفرعية بتحديد أول موقع وأخر موقع في السلسة ٥ – ٨ في السلسلة "This is the first program" سلسلة فرعية ( ) .
٥- الادخال (Insertion) حيث يتم إدخال رمز أو سلسلة فرعية وذلك بتحديد الموقع في السلسلة الاصلية ، فمثلاً لإدخال "First" في السلسلة " This is the program " ومن الموقع ١٢ تظهر السلسلة الجديدة كمايلي : " This is the first program ".
٦- الحذف (Deletion) حيث يحدد الرمز أو السلسلة الفرعية المراد حذفها من السلسلة الأصلية وذلك بمعرفة أول موقع للسلسلة الفرعية المراد حذفها.

تخزين السلاسل " String Storage " :-

وسوف نستعرض ثلاث طرق لتخزين السلاسل :

١- طريقة السلاسل ذات الطول الثابت .
٢- طريقة استخدام الجداول المفهرسة.
٣- طريقة القوائم المتصلة.

١- طريقة السلاسل ذات الطول الثابت :Fixed length string method

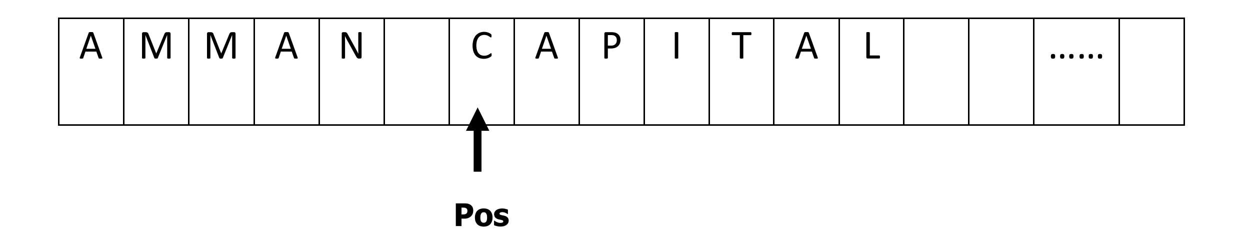
في هذه الطريقة يحجز لكل سلسلة عدد محدد من المواقع في الذاكرة وتخزن رموز السلسلة في هذه المواقع وفي حالة وجود مواقع فائضة فإنها تخزن بالفراغ ولا يمكن استخدامها من قبل سلسلة أخرى ، وتعتبر هذه الفراغات إحدى مساوئ هذه الطريقة إذا أنها تؤدي إلى مضيعة الذاكرة وعدم إستغلالها بصورة مثالية .

عملية الإضافة : Insertion

عند إجراء عملية إضافة سلسلة في سلسلة أخرى نتبع الخطوات التالية :

١- يجب أن لا يزيد طول السلسلة المضافة والسلسلة الأصلية على طول السلسلة الأصلية.
٢- يحدد الموقع " البداية " المراد إجراء عملية الإضافة عنده .
٣- تتم إزاحة عناصر السلسلة الأصلية لليمين وذلك لإيجاد مكان لعناصر السلسلة المراد إضافتها .

لو فرضنا أننا نريد إدخال السلسلة ( ) في السلسلة التالية ومن الموقع السابع :



فإنه لابد من إجراء مايلي :

١- التحقق من إمكانية الإضافة (13 +2 <= 30) .

تنفيذ الإضافه

٢- اجعل i = 13 .
٣- إزاحة العنصر المشار اليه بـ i مسافه مساوية لطول السلسلة المضافة.

```
Str(i+2):=Mast(i);
```

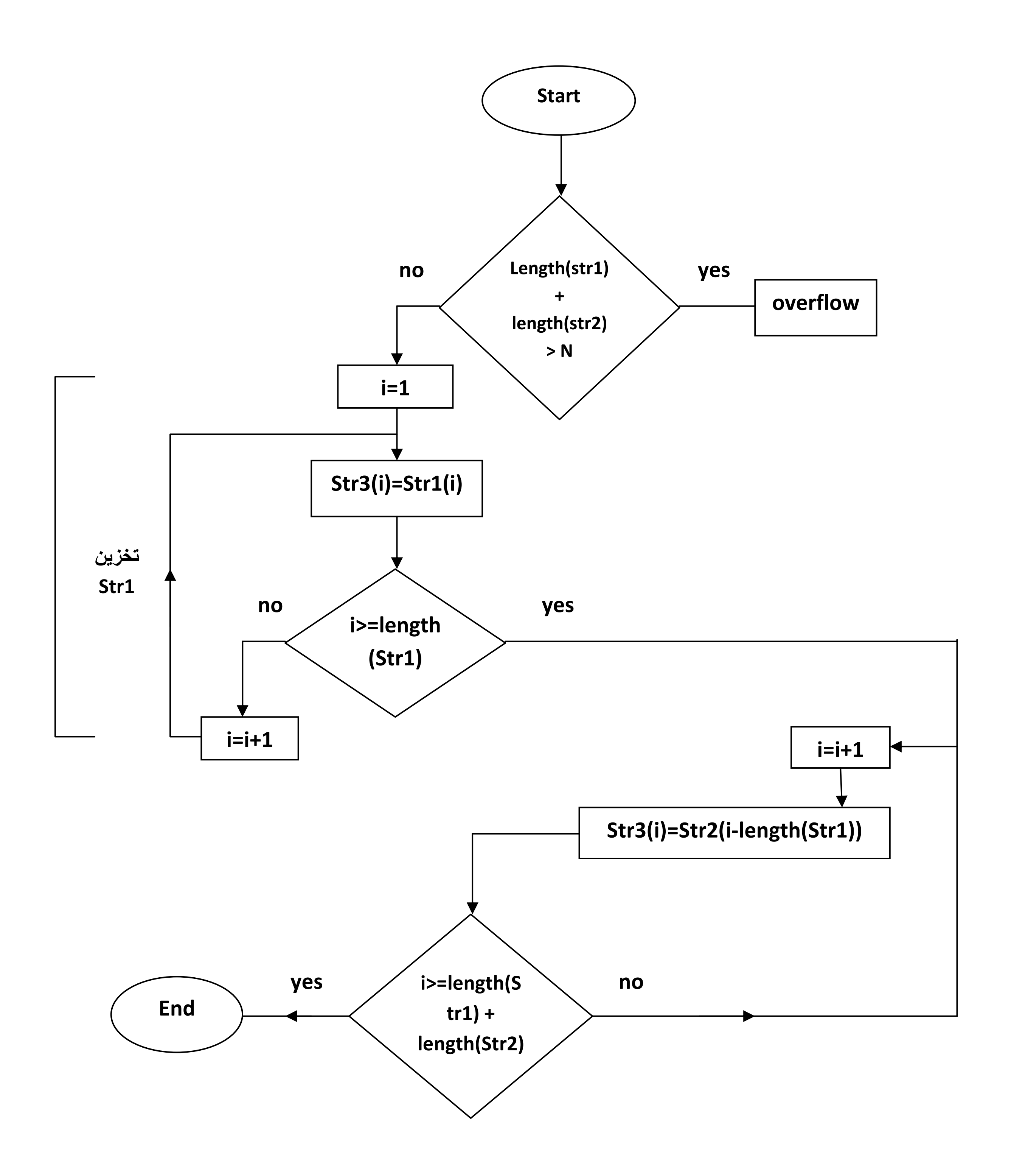
٤- طرح واحد من i (i = i-1) وتكرر الخطوة (٣) ما دامت (i > = pos).
٥- تمثل الخطوات من (١ – ٤) عملية الازاحة ، وبعد الانتهاء منها تنفذ عملية الاضافة (إضافة عناصر السلسلة الجديدة ويبدأ من الموقع (Pos)).

عملية دمج سلسلتين : Concatenation

تتم عملية دمج سلسلتين في سلسلة ثالثة حسب الخطوات التالية :

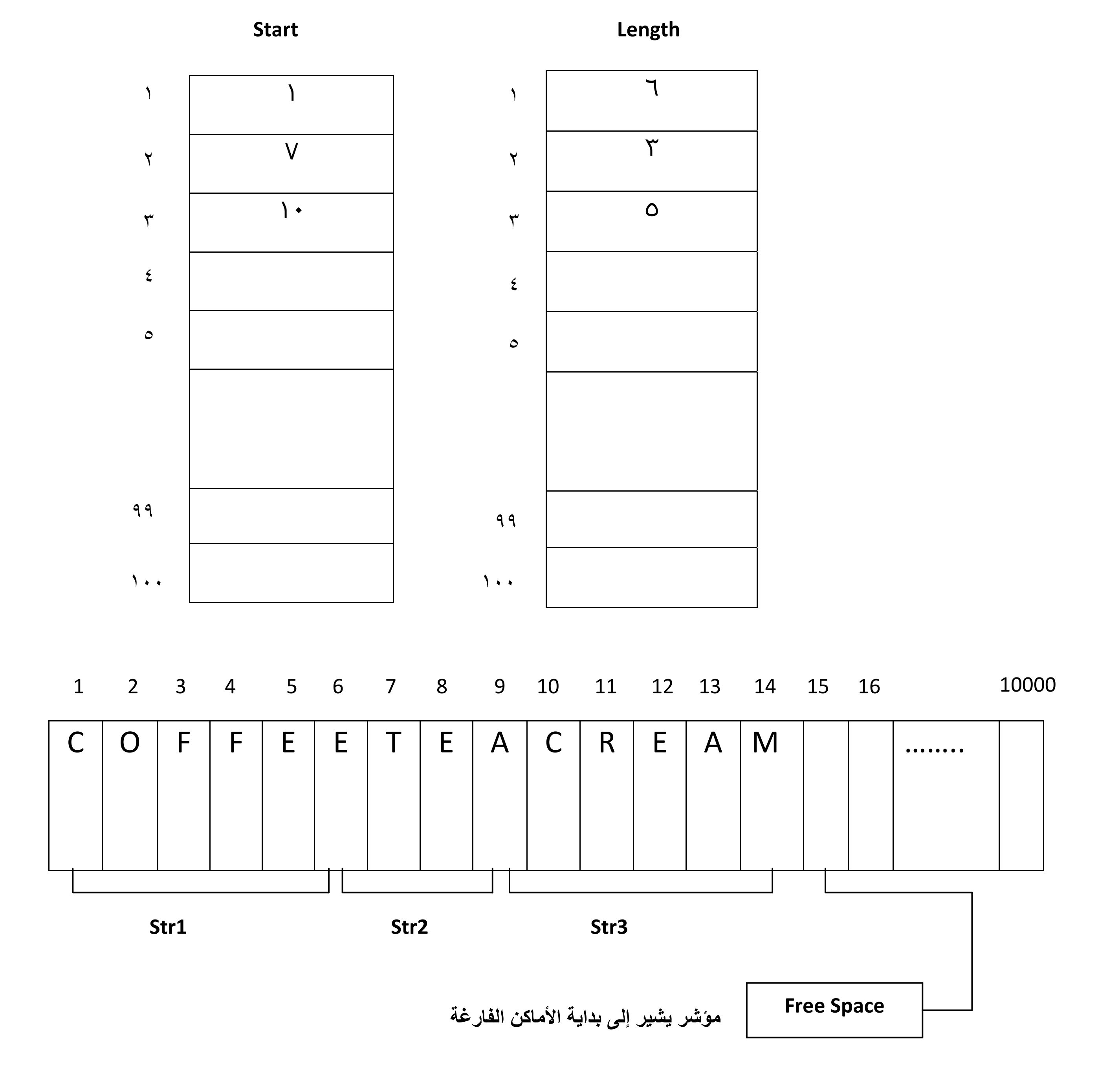
١- يتم فحص مجموع طول السلسلتين فاذا كان أكبر من (N) (طول السلسة الثالثة) فمعنى هذا آن المكان المخصص لا يفي وإلا نفذ الخطوات التالية :
٢- خزن عناصر السلسلة الأولى.
٣- خزن عناصر السلسله الثانية .

ويمكن تمثيل هذه العملية بمخطط سير العمليات التالية حيث أن السلسلة الأولى Str1 ، السلسله الثانيه str2 ، السلسلة الثالثة Str3 .



Index table/work space method: 2- طريقة إستخدام الجداول المفهرسة

في هذه الطريقة يتم بناء جدول البداية (Start table) يحدد فيه بداية كل سلسلة كما يبنى جدول الأطوال (length table) يحدد في كل مدخل فيه طول كل سلسلة من السلاسل في مصفوفة المساحه(work space) والشكل التالي يبين هذه الجداول :



٣- طريقة القوائم المتصله :Lenked lists method

يمكن أن تمثل السلاسل بواسطة القوائم الثنائية فكل حرف في السلسلة يوضع في عقدة لها مؤشرين مؤشر أمامي ومؤشر خلف ويتضح ذلك عند التطرق لهذا الهيكل من هياكل البيانات .

(٢-١) المصفوفات Arrays :-

هي عباره عن هيكل من هياكل البيانات ، وهي عباره عن مجموعه منتهيه ومرتبه من العناصر المتجانسه .

والبيانات التي تخزن داخل المصفوفه لها شروط منها:-

١. أن تكون منتهيه ، أي أن يكون للمصفوفه حجم معين والبيانات تكون محدده وإذا أردنا أن نتعامل مع المصفوفات فلا بد من تحديد حجم المصفوفه ويتم ذلك بواسطة المستخدم ، ومن سلبياتها أنه إذا تم حجز مساحة معينة للمصفوفة ولم تستغل هذه المساحه بشكل كامل فلا يمكن استغلال هذه المساحه بواسطة متغير آخر وتبقى هذه المساحه فارغه.
٢. أن البياتات تكون مرتبه مثل x1,x2,x3,... .
٣. أن تكون المصفوفه متجانسه ، ونقصد بالتجانس أنه لايمكن جمع بيانات مختلفه في أنواعها وجعلها في مصفوفة واحده ، وبالتالي لابد أن تكون البيانات من نوع واحد مثل Integer,real,string,...

نعني بقولنا منتهية (Finite) أي أن لها عدد محدود من العناصر ، ومرتبة (ordered) أي أن عناصرها مرتبة الأول فالثاني فالثالث ... وهكذا ، ومتجانسة (homogenous) أي أن جميع عناصرها من نفس النوع (Type) .

أنواع المصفوفات Types :-

لها ثلاثة أنواع :-

١- مصفوفة أحادية الأبعاد .
٢- مصفوفة ثنائية الأبعاد . وهما الأكثر إستخداماً (١ ، ٢) .
٣- مصفوفة متعددة الأبعاد (Multidimensional array) .

الصيغة Syntax :-

هنالك طريقتان للتصريح عن المصفوفات :-

١- أن تقوم بتعريف المصفوفة على أنها نوع بيانات Data type ، فلا بد أن يتم حجز مساحة لها داخل الذاكره ومن ثم نقوم بتعريف هذه البيانات (real , char, Jnteger) ، ويمكن أن تكون هنالك بيانات لا يوجد تعريف لها ولكن هنالك مواقع محددة في شكل البرنامج العام

يتم فيها التعريف عن أنواع البيا نات الجديدة في لغة pascal وذلك بواسطة الكلمة المحجوزة Type .

```
Type
Arrayname = Array [index] of componettype;
```

- **Arrayname** : إسم المصفوفة ، عندما نستخدم إسم المصفوفة تصبح إسم المصفوفة كأنها نوع من أنواع البيانات المعرفة بلغة الباسكال .
- **Index** : وهو عدد عناصر المصفوفة ولابد أن يكون من النوع الترتيبي Baslean , char , integer .
- **Data type or componettype**: هو نوع البيانات ، ولابد أن يكون نوع البيانات من الأنواع الأساسية المعرّفة في لغة الباسكال مثل Char , int ... وإذا لم تكن معرفة فلا بد أن يتم نعريفها وذلك بواسطة الكلمة المحجوزة Type كما سبق ذكره ، وإذا كان نوع البيانات من النوع Integer يحجز لها في الذاكرة 10 bytes .

Example:-

```
Type
Matrix = array [1..5] Of integer;
 var a , b, c : Matrix;
```

- **Matrix :** مصفوفة عدد عناصرها 5 عناصر .
- **a :** مصفوفة نوع بياناتها من النوع integer وتحتوي على 5 عناصر .
- إسم المصفوفه هو نوع البيانات التي يتعرف عليها البرنامج في لغة Pascal لتعريف المصفوفات a,b,c وهذه الطريقة تستخدم إذا كان هنالك عدد من المصفوفات متشابهة في البيانات ومتساوية في الأحجام وفي منطقة التعريف var أكتب متغيرات على حسب عدد المصفوفات .

٢- تستخدم هذه الطريقة الثانية إذا كان هنالك مصفوفة واحدة ففي هذه الحالة أعرفها في منطقة التعريف var وكذلك تستخدم هذه الطريقة إذا كان هنالك عدد من المصفوفات غير متشابهة .

```
Var

Arrayname : array [index] of data type;
```

Example:-

```
Var

X : array [index] of  integer;
```

في هذه الحالة الـarrayname يكون variable .

أنواع المصفوفات :-

(١) المصفوفة أحادية الأبعاد :-

- سنتعرف على كيفية تخزين البيانات داخل المصفوفة وكيفية الوصول إلى البيانات لإجراء عمليات المعالجة ومن ثم كتابة عدد من البرامج .
- إذا أردت أن أخزن Data في المصفوفة أو أخرج Data أستخدم For loop ، وإن عملية التخزين في المصفوفة عملية قائمة بذاتها وعملية الاسترجاع في المصفوفة عملية قائمة بذاتها فبذلك نحتاج حلقة للتخزين وحلقة للاسترجاع ، وبعد المصفوفة هو الذي يحدد عدد الحلقات ، وفي المصفوفة أحادية الأبعاد أستخدم حلقة واحدة للتخزين وحلقة ثانية للإخراج أو المعالجة أو الإسترجاع وكذلك أعرف متغير للإستخدام كعداد Counter ، والحلقة الخارجية تمثل عدد الصفوف row والحلقة الداخلية تمثل عدد الأعمدة columns .
- إذاً إذا طرح عليك هذا السؤال فما هي اجابتك : كيف تخزن عناصر المصفوفة ؟

فالجواب هو :بإستخدام حلقة For loop .

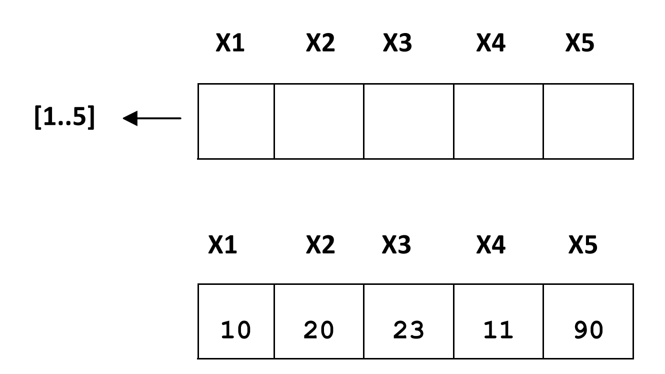
Simple Example:-

```
program arrays;
var x:array[1..5]of integer;
i:integer;
begin
writeln('enter the data of array x....');
for i:=1 to 5 do
begin
write('enter element x[',i,']: ');
readln(x[i]);
end;
writeln('the data of array x is:');
for i:=1 to 5 do
```

```
write(x[i],' ');
readln;
end.
```

Output:-

```
enter the data of array x....
enter element x[1]: 10
enter element x[2]: 20
enter element x[3]: 23
enter element x[4]: 11
enter element x[5]: 90
the data of array x is:
10 20 23 11 90
enter the data of array x....
```



إذا قمنا بتعديل البرنامج السابق وجعلنا عداد حلقة for loop عام ولكنه ينحصر في المدى [1..5] فإنه يكون كالتالي :-

```
program arrays;
var x:array[1..5]of integer;
i,n:integer;
begin
writeln('enter the elements number of array x....');
readln(n);
```

```
for i:=1 to n do
begin
write('enter element x[',i,']: ');
readln(x[i]);
end;
writeln('the data of array x is:');
for i:=1 to n do
write(x[i],' ');
readln;
end.
```

Output:-

```
enter the elements number of array x....
٣
enter element x[1]: 12
enter element x[2]: 33
enter element x[3]: 56
the data of array x is:
٥٦ ٣٣ ١٢
```

Example:-

أكتب برنامج يعمل على تخزين عناصر المصفوفة X ويقوم بجمع العناصر وطباعتها على شاشة التنفيذ.

```
program arrays;
var x:array[1..5]of integer;
i:integer;  sum:longint;
begin
writeln('enter the data of array x....');
sum:=0;
for i:=1 to 5 do
begin
```

```
write('enter element x[',i,']: ');
readln(x[i]);
sum:=sum+x[i];
end;
writeln('sumation x= ',sum);
readln;
end.
```

Output:-

```
enter the data of array x....
enter element x[1]: 10
enter element x[2]: 2
enter element x[3]: 1
enter element x[4]: 20
enter element x[5]: 3
sumation x= 36
```

في هذه الحاله يتم تخزين العنصر الأول ثم يجمع لل sum وهذه الطريقة هي الأفضل ، أما الطريقة الثانية فهي كالتالي:-

```
Sum:=0;
For i:=1 to 5 do
Readln(x[i]);
For i:=1 to 5 do
Sum:=sum+x[i];
Writeln('sumation x:',sum);
```

في هذه الحالة يقوم بتخزين البيانات كلها ثم يقوم بعملية الجمع .

أما إذا كان المطلوب إجراء العملية التالية فسيكون كالتالي:-

$$\text{Sum}=\sum_{i=1}^{5} xi^2$$

```
Sum:=0;
For i:=1 to 5 do
```

```
Begin
Readln(x[i]);
Sum:=sum+sqr(x[i]);
End.
```

تمرين(١):-

(١) أكتب برنامج يخزن عناصر مصفوفة ثم يوجد عملية الإنحراف المعياري ثم يطبع عناصر المصفوفة وقمية الإنحراف $sd=\frac{\sqrt{\sum_{i=1}^{n}(\text{vi}-\bar{\text{v}})^2}}{n}$.

(٢) أكتب برنامج يعمل على تخزين عناصر المصفوفات A,B ثم يقوم البرنامج بجمع هذه المصفوفات وتخزينها على المصفوفة C ويقوم البرنامج كذلك بإظهار بيانات المصفوفات الثلاثه على شاشة التنفيذ علماً بأن المصفوفات الثلاثة أحادية الأبعاد.

حل التمرين:-

(1)
```
program one;
var s:array[1..5] of integer;
j:integer; sumation1:longint;
sumation2,avg,sd:real;
begin
sumation1:=0; sumation2:=0;
writeln('Enter the data of array s:');
for j:=1 to 5 do
begin
write('Enter element s',j,' ');
readln(s[j]);
sumation1:=sumation1+s[j];
end;
avg:=sumation1/5;
for j:=1 to 5 do
sumation2:=sumation2+s[j]-avg;
sd:=sqrt(sqr(sumation2))/5;
writeln;
writeln('The data of array s is:');
for j:=1 to 5 do
```

```
write(s[j],' ');
writeln;
writeln('The standard deviation is :',sd);
readln;
end.
```

Output:-

```
Enter the data of array s:

Enter element s1 20

Enter element s2 10

Enter element s3 12

Enter element s4 44

Enter element s5 11

The data of array s is:

20 10 12 44 11

The standard deviation is : 5.8207660913E-12
```

(2) program two;

```
type
xy=array[1..10] of integer;
var a,b,c:xy;
n,j:integer;
begin
writeln('Enter the data of array xy:');
writeln('Enter n number:');
readln(n);
for j:=1 to n do
begin
write('Enter the element a',j,'  ');
readln(a[j]);
end;
writeln;
for j:=1 to n do
```

```
begin
write('Enter the element b',j,' ');
readln(b[j]);
end;
for j:=1 to n do
c[j]:=a[j]+b[j];
writeln('The data of array a is:');
for j:=1 to n do
write(a[j],' ');
writeln;
writeln('The array of b is:');
for j:=1 to n do
write(b[j],' ');
writeln;
writeln('The sumation of arrays a,b is:');
for j:=1 to n do
write(c[j],' ');
readln;
end.
```

Output:-

```
Enter the data of array xy:
Enter n number:
3
Enter the element a1  12
Enter the element a2  23
Enter the element a3  9

Enter the element b1 33
Enter the element b2 11
Enter the element b3 41
The data of array a is:
12 23 9
The array of b is:
33 11 41
The sumation of arrays a,b is:
```

```
45 34 50
```

(٢) المصفوفة ثنائية الأبعاد :-

حتى نستطيع أن نجري عملية معالجة على data معينة فلا بد من أن نكون خزنا هذه الـ data على ذاكرة الحاسوب على وسط تخزيني ويتم ذلك بطريقة فيزيائية .

والوسائط التخزينية نوعان :

١- وسائط تخزين أولية . ٢- وسائط تخزين ثانوية.

الخطوات :-

١- تخصيص مساحة لهذا الهيكل .
٢- قبل المعالجة نخزن هذه البيانات على الذاكرة .
٣- لابد أن نحدد العنوان للـ Data حتى تجرى عملية المعالجة .

- أولاً لابد أن نحجز مساحة على ذاكرة الحاسوب وذلك يتم بالتصريح عن المصفوفات وذلك يتم عن طريقة التعريف Type :

```
type
matrix=array [1..30,1..30] of integer;
var  x,y,z:matrix;
```

- بتخزين المتغيرات نكون قد ضمنا مساحة داخل ذاكرة الحاسوب وكذلك العناوين ، وتتعامل مع x على أنها مصفوفة عدد عناصرها ٩٠ عنصر ونوع بياناتها من النوع integer .
- بهذا يكون لكل عنصر مساحة وعنوان سواء أكانت هذه العناصر في منطقة واحدة أو في مناطق متعددة .
- متى ما أردت أن أتعامل مع أي مصفوفة لا أستطيع أن أصل إلى عناصرها إلا بإستخدام forloop وتكون أعداد الحلقات على حسب بعد المصفوفه.
- في هذه الحالة نحتاج إلى حلقتين:

١- حلقة خارجية وتحدد عدد الصفوف.
٢- حلقة داخلية وتحدد عدد الأعمدة.

سؤال: كيف يتم تخزين عناصر المصفوفة ثنائية الأبعاد ؟؟

أولاً يتم تخزين بيانات الصف الأول كاملة ثم بيانات الصف الثاني إلى آخر صف وبذلك عندما يكون العداد = ١ فإننا نكون نتعامل مع الصف الأول

```
for i:=1 to 3 do
```

```
    for j:=1 to 5 do
```

وهذا يعني أنه يوجد ٣ صفوف و ٥ أعمدة .

Example:-

نريد تخزين عناصر المصفوفة x ثم عناصر المصفوفة y ثم نقوم بعملية الجمع للمصفوفتين وإظهارها على المصفوفة z .

```
program three;
type
matrix=array[1..30,1..30] of integer;
var x,y,z:matrix;
i,j,k,m1,m2,n1,n2:integer;
begin
writeln('enter the demintion of array x:','m1<=30,n1<=30');
readln(m1,n1);
writeln('enter the data of array x:');
for i:=1 to m1 do
for j:=1 to n1 do
begin
write('enter element x',i,j,' ');
readln(x[i,j]);
end;
writeln('enter the demintion of array y:','m2<=30,n2<=30');
readln(m2,n2);
writeln('enter the data of array y:');
for i:=1 to m2 do
for j:=1 to n2 do
begin
write('enter element y',i,j,' ');
readln(y[i,j]);
end;
if n1=m2 then
begin
for i:=1 to m1 do
for j:=1 to n2 do
begin
z[i,j]:=0;
for k:=1 to m2 do
```

```
z[i,j]:=z[i,j]+x[i,k]*y[k,j];
end;
writeln;
writeln('the data of array x is:');
for i:=1 to m1 do
begin
for j:=1 to n1 do
write(x[i,j],' ');
writeln;
end;
writeln;
writeln('the data of array y is:');
for i:=1 to m2 do
begin
for j:=1 to n2 do
write(y[i,j],' ');
writeln;
end;
writeln;
writeln('the data of array z is:');
for i:=1 to m1 do
begin
for j:=1 to n2 do
write(z[i,j],' ');
writeln;
end;
end;
readln;
end.
```

Output:-

```
enter the demintion of array x:m1<=30,n1<=30
2
2
enter the data of array x:
enter element x11 12
```

```
enter element x12 1
enter element x21 3
enter element x22 5
enter the demintion of array y:m2<=30,n2<=30
2
2
enter the data of array y:
enter element y11 4
enter element y12 7
enter element y21 1
enter element y22 9

the data of array x is:
12 1
3  5

the data of array y is:
4 7
1 9

the data of array z is:
49 93
17 66
```

- عند إجراء عملية الجمع وتخزين الناتج على المصفوفة z أو المصفوفة الثالثة يكون عدد صفوف المصفوفة الثالثة يساوي عدد صفوف المصفوفة الأولى وعدد أعمدة المصفوفة الثالثة يساوي عدد أعمدة المصفوفة الثانية .
- من الممكن أن يكون عدد الصفوف من ١ ←— ٣٠ صف ولايتجاوز العدد المحدد وذلك من سلبيات المصفوفات وهو أن مساحة المصفوفة لابد أن تحدد أولاً وأنه لايمكن تعدي الماحة المخزنة وأنه عندما نريد تخزين بيانات المصفوفة ثنائية الأبعاد فلابد من تحديد عدد الصفوف وعدد الأعمدة.
- من الممكن أن يتم تحديد عدد العناصر على أنه ثابت const كالتالي:-

```
Const  n=3;
Type
Matrix = array [1..n,1..n] of integer;
```

Example:-

```
program four;
```

```
type
matrix=array[1..20,1..20] of integer;
var x,y,z:matrix;
no,i,j,m1,m2,n1,n2:integer;
procedure input(var x,y:matrix);
begin
writeln('enter demintion array x:,m1<=20,n1<=20');
readln(m1,n1);
writeln('enter the data of array x:');
for i:=1 to m1 do
for j:=1 to n1 do
begin
write('enter element x',i,j,' ');
readln(x[i,j]);
end;
writeln('enter demintion array y:,m2<=20,n2<=20');
readln(m2,n2);
writeln('enter the data of array y:');
for i:=1 to m2 do
for j:=1 to n2 do
begin
write('enter element y',i,j,' ');
readln(y[i,j]);
end;
end;
procedure add(var x,y,z:matrix);
begin
for i:=1 to m1 do
for j:=1 to n2 do
z[i,j]:=x[i,j]+y[i,j];
end;
procedure display(x,y,z:matrix);
begin
writeln('the data of array x is:');
for i:=1 to m1 do
begin
```

```
for j:=1 to n1 do
write(x[i,j],' ');
writeln;
end;
writeln;
writeln('the data of array y is:');
for i:=1 to m2 do
begin
for j:=1 to n2 do
write(y[i,j],' ');
writeln;
end;
writeln;
writeln('the sumation of arrays x,y is:');
for i:=1 to m1 do
begin
for j:=1 to n2 do
write(z[i,j],' ');
writeln;
end;
end;
begin
writeln('enter 1 ---> input , 2 ---> add , 3 ---> display , 4 ---> exit');
repeat
writeln('enter your choices');
readln(no);
case no of
1:input(x,y);
2:add(x,y,z);
3:display(x,y,z);
end;
until (no=4);
readln;
end.
```

Output:-

```
enter 1 ---> input , 2 ---> add , 3 ---> display , 4 ---> exit

enter your choices

1

enter demintion array x:,m1<=20,n1<=20

2

2

enter the data of array x:

enter element x11 12

enter element x12 4

enter element x21 11

enter element x22 2

enter demintion array y:,m2<=20,n2<=20

2

2

enter the data of array y:

enter element y11 9

enter element y12 4

enter element y21 4

enter element y22 2

enter your choices

2

enter your choices

3

the data of array x is:

12 4

11 2


the data of array y is:

9 4
```

```
4 2

the sumation of arrays x,y is:
21 8
15 4
enter your choices
4
```

إجراء عملية الضرب:-

١- لاحظ أننا نقوم بإدخال x و y .
٢- لابد أن نأخذ في الإعتبار أن عدد أعمدة المصفوفة الأولى تساوي عدد صفوف المصفوفة الثانية -شرط ضرب المصفوفات- .
٣- لابد أن يتم إستخدام if statement ويكون كالتالي: if n1= m2 then .

```
if n1=m1 then
Begin
for i:=1 to m1 do
for j:=1 to n2 do
begin
Z[i,j]:=0;
for k:=1 to m2 do
Z[i,j]:=z[i,j]+x[i,k]*g[k,j];
end;
end;
else
writeln('enter n1=m2');
```

- عندما يكون عداد الحلقة الداخلية قيمته (١) هذا يعني أنه في العمود الأول وعندما يكون قيمة العداد (٢) هذا يعني أنه في العمود الثاني وهكذا ، وعندما تكون قيمة الحلقة الخارجية (١) تنفذ الحلقة الداخلية كاملة .

- عملية الضرب أو الجمع تعني أوتسمى عمليات معالجة.
- بالنسبة لعداد الحلقة k :

يتم أخذ العنصر الأول في الصف الأول وضربه في العنصر الأول في العمود الأول المقابل له وجمعهما إلى العنصر الثاني في الصف الأول وضربه في العنصر الأول في العمود الثاني المقابل له وجمعهما إلى العنصر الثالث في الصف الأول وضربه في العنصر الأول في العمود الثالث ووضع قيمتهم في المصفوفة الجديدة في الصف الأول في العمود الأول وهكذا إلى بقية الصفوف والأعمدة.

Example:-

```
program five;
type
matrix=array[1..20,1..20] of integer;
var x,y,z:matrix;
no,i,j,k,m1,m2,n1,n2:integer;
procedure input(var x,y:matrix);
begin
writeln('enter demintion array x:,m1<=20,n1<=20');
readln(m1,n1);
writeln('enter the data of array x:');
for i:=1 to m1 do
for j:=1 to n1 do
begin
write('enter element x',i,j,' ');
readln(x[i,j]);
end;
writeln('enter demintion array y:,m2<=20,n2<=20');
readln(m2,n2);
writeln('enter the data of array y:');
for i:=1 to m2 do
for j:=1 to n2 do
begin
write('enter element y',i,j,' ');
readln(y[i,j]);
end;
end;
```

```
procedure mult(var x,y,z:matrix);
begin
if n1=m2 then
begin
for i:=1 to m1 do
for j:=1 to n2 do
begin
z[i,j]:=0;
for k:=1 to m2 do
z[i,j]:=z[i,j]+x[i,k]*y[k,j];
end;
end
else writeln('enter n1=m2');
end;
procedure display(x,y,z:matrix);
begin
writeln('the data of array x is:');
for i:=1 to m1 do
begin
for j:=1 to n1 do
write(x[i,j],' ');
writeln;
end;
writeln;
writeln('the data of array y is:');
for i:=1 to m2 do
begin
for j:=1 to n2 do
write(y[i,j],' ');
writeln;
end;
writeln;
writeln('the multiplication of arrays x,y is:');
for i:=1 to m1 do
begin
for j:=1 to n2 do
```

```
write(z[i,j],' ');
writeln;
end;
end;
begin
writeln('enter 1 ---> input , 2 ---> mult , 3 ---> display , 4 --->
exit');
repeat
writeln('enter your choices');
readln(no);
case no of
1:input(x,y);
2:mult(x,y,z);
3:display(x,y,z);
end;
until (no=4);
readln;
end.
```

Output:-

```
enter 1 ---> input , 2 ---> mult , 3 ---> display , 4 ---> exit
enter your choices
1
enter demintion array x:,m1<=20,n1<=20
2
2
enter the data of array x:
enter element x11 2
enter element x12 3
enter element x21 4
enter element x22 5
enter demintion array y:,m2<=20,n2<=20
2
```

```
2
enter the data of array y:
enter element y11 6
enter element y12 7
enter element y21 8
enter element y22 9
enter your choices
2
enter your choices
3
the data of array x is:
2 3
4 5

the data of array y is:
6 7
8 9

the multiplication of arrays x,y is:
36 41
64 73
enter your choices
4
```

****

الوحدة الثانية

السجلات

Records

السجلات Records

- السجل هو هيكل بيانات ولكل هيكل طريقة معينة للتعامل معه .
- السجلات تشابه المصفوفات في أن البيانات التي تخزن في السجلات لها علاقة مع بعضها البعض وهنا نتعامل مع الحقول ، والحقل هو الذي تخزن فيه البيانات داخل ذاكرة الحاسوب .
- السجلات تمتاز عن المصفوفات في أنها يمكن أن تخزن بيانات مختلفة ، يمكن أن نعترف كل حقل من نوع بيانات مختلف ويمكن تجمع بيانات مختلفة الأنواع ونضعها في سجل واحد .
- والسجل يتكون من عدد من الحقول .

التصريح عن السجلات:-

- السجلات تحتاج إلى هيكل بيانات أكبر للمعالجة وفي هذه الحالة لابد أن نعرّف نوع بيانات جديد يسمي (record) ونستخدم للتصريح عن السجلات طريقة (Type) .

```
type
recordName=record
filed1:dataType;
filed2:dataType;
filed3:dataType;

   ↓

filedn:dataType;
end;
var variableName:recordName;
```

أي في منطقة var نعرف المتغير من النوع recordname .

- السجلات (record) تشبه المصفوفات في كونها مجموعة من البيانات المتعلقة ببعضها البعض ، وتختلف عنها في إمكانية إحتوائها على بيانات مختلفة في النوع .

Example:-

```
type
student=record
name:string;
no:integer;
place:string;
age:integer;
end;
var st:student;
```

هنا نتعامل مع st على أنه سجل وفيه خمسة حقول.

كيف يتم تخزين البيانات على السجلات؟؟

هنالك طريقتان:-

١- طريقة dot :-

Example: `recordname.fieldname;`

Recordname: إسـم المتغير الذي يمثل إسـم السـجل .

Fieldname: إسـم الحقل الذي نريد أن نتعامل معه ، وعملية الإخراج تكون بنفس الطريقة.

٢- طريقة with :-

إذا كان هنالك سـجل به عدد كبير من الحقول فعملية التكرار لإسـم السـجل تكون مملة ولكن من الممكن بإسـتخدام عبارة with أن يتم التعامل مع كل الحقول من دون كتابة إسـم المتغير وإنما يتم التعامل مع الحقول مباشـرة.

Example:-

أكتب برنامج يقوم بتخزين بيانات الطالب ، علماً بأن بيانات الطالب هي : الإسـم والرقم والمكان والعمر والنوع مستخدما طريقة dot مرة وطريقة with مرة ثانية.

Dot:-

```
program six;
type
student=record
name:string;
no:integer;
place:string;
age:integer;
type_st:string;
end;
var st:student;
begin
writeln('enter the data of student:');
writeln('enter student name:');
readln(st.name);
writeln('enter student number:');
```

```
readln(st.no);
writeln('enter student place:');
readln(st.place);
writeln('enter student age:');
readln(st.age);
writeln('enter student type:');
readln(st.type_st);
writeln;
writeln('the data of student is:');
writeln('name:',st.name);
writeln('number:',st.no);
writeln('place:',st.place);
writeln('age:',st.age);
writeln('type_student:',st.type_st);
readln;
end.
```

Output:-

```
enter the data of student:
enter student name:
safiya
enter student number:
1
enter student place:
khartoum
enter student age:
19
enter student type:
female

the data of student is:
name:safiya
number:1
place:khartoum
age:19
type_student:female
```

with:-

```
program seven;
type
student=record
name:string;
no:integer;
place:string;
end;
var st:student;
begin
writeln('enter the data of student:');
with st do
begin
write('enter student name:');
readln(name);
write('enter student number:');
readln(no);
write('enter student place:');
readln(place);
end;
writeln;
writeln('the data of student is:');
with st do
begin
writeln('name:',name);
writeln('number:',no);
writeln('place:',place);
end;
readln;
end.
```

Output:-

```
enter the data of student:
enter student name:safiya
enter student number:4
enter student place:baghadad
```

```
the data of student is:
name:safiya
number:4
place:baghadad
```

Example:-

أكتب برنامج يقوم بتخزين بيانات ٣ طلاب وطباعة بياناتهم على شاشة التنفيذ.

```
program eight;
type
student=record
name:string;
no:integer;
place:string;
end;
var st:array[1..3] of student;
i:integer;
begin
writeln('enter the data of student:');
for i:=1 to 3 do
with st[i] do
begin
write('enter name:',' ');
readln(name);
write('enter number:',' ');
readln(no);
write('enter place:',' ');
readln(place);
end;
writeln;
writeln('the data of student is:');
writeln('name','    ','number','    ','place','    ');
for i:=1 to 3 do
with st[i] do
begin
writeln(name,'    ',no,'    ',place,'    ');
```

```
end;
readln;
end.
```

Output:-

```
enter the data of student:
enter name: banan
enter number: 3
enter place: tunis
enter name: haneen
enter number: 1
enter place: monester
enter name: yosof
enter number: 6
enter place: madany

the data of student is:
name     number     place
banan      3        tunis
haneen     1        monester
yosof      6        madany
```

Example:-

أكتب برنامج يعمل على تخزين سجلات ٣ موظفين ثم يقوم البرنامج بطباعة بيانات آخر موظف على شاشة التنفيذ مع إستغلال مساحة التخزين ومراعاتها.

```
program nine;
type
employee=record
name:string;
no:integer;
age:integer;
end;
var emp:employee;
i:integer;
begin
```

```
writeln('enter the data of employee:');
for i:=1 to 3 do
with emp do
begin
write('enter name:',' ');
readln(name);
write('enter no:',' ');
readln(no);
write('enter age:',' ');
readln(age);
end;
writeln;
writeln('the data of employee is:');
with emp do
begin
writeln('name:',name);
writeln('number:',no);
writeln('age:',age);
end;
readln;
end.
```

Output:-

```
enter the data of employee:
enter name: najeh
enter no: 12
enter age: 50
enter name: fatima
enter no: 3
enter age: 21
enter name: riad
enter no: 6
enter age: 41

the data of employee is:
name:riad
```

```
number:6
age:41
```

في هذا المثال تم حجز مساحة تسع لسجل واحد فقط وليس لثلاثة سجلات ، أما إذا كان مطلوب تخزين بيانات ثلاثة موظفين وطباعة بياناتهم لابد أن نستخدم مصفوفة كالتالي:

```
var emp:array [1..3] of employee;
```

السجلات المتداخلة : nested record

Example:-

```
program ten;
type
brith=record
day:integer;
month:integer;
year:integer;
end;
student=record
name:string;
no:integer;
brithday:brith;
end;
var st:student;
begin
writeln('enter the data of student:');
write('enter name :');
readln(st.name);
write('enter no:');
readln(st.no);
write('enter brith day:');
write('day:');
readln(st.brithday.day);
write('month:');
readln(st.brithday.month);
write('year:');
readln(st.brithday.year);
```

```
writeln;
writeln('the data of student is:');
writeln('name:',st.name);
writeln('number:',st.no);
writeln('brithday:');
writeln('day:',st.brithday.day);
writeln('month:',st.brithday.month);
writeln('year:',st.brithday.year);
readln;
end.
```

Output:-

```
enter the data of student:
enter name :safiya
enter no:12
enter brith day:day:13
month:12
year:1991

the data of student is:
name:safiya
number:12
brithday:
day:13
month:12
year:1991
```

تمرين(٢):-

اكتب برنامج يعمل على تخزين سجلات ٥ موظفين على المصفوفة (emp) علماً بأن بيانات الموظف هي الإسم والرقم والمرتب وتاريخ الميلاد (عبارة عن سجل حقوله هي اليوم والشهر والسنة) ، ويقوم البرنامج بطباعة سجلات الموظفين على شاشة التنفيذ .

حل التمرين:-

```
program eleven;
type
brith=record
day:integer;
```

```
month:integer;
year:integer;
end;
employee=record
name:string;
no:integer;
salary:real;
brithday:brith;
end;
var emp:array[1..5] of employee;
i:integer;
begin
writeln('the data of employee is:');
for i:=1 to 5 do
with emp[i] do
begin
write('enter name:',' ');
readln(name);
write('enter number:',' ');
readln(no);
writeln('enter brith day:');
write('day:');
readln(brithday.day);
write('month:');
readln(brithday.month);
write('year:');
readln(brithday.year);
end;
writeln;
writeln('the data of employee is:');
writeln('name','    ','number','    ','brithday.day','
','brithday.month','    ','brithday.year','    ');
for i:=1 to 5 do
with emp[i] do
begin
```

```
writeln(name,'          ',no,'              ',brithday.day,'
',brithday.month,'                ',brithday.year,'       ');
end;
readln;
end.
```

Output:-

```
the data of employee is:
enter name: kamal
enter number: 1
enter brith day:
day:13
month:3
year:1984
enter name: fady
enter number: 2
enter brith day:
day:19
month:4
year:1995
enter name: banan
enter number: 3
enter brith day:
day:28
month:9
year:2007
enter name: yosof
enter number: 5
enter brith day:
```

```
day:82

month:10

year:2009

enter name: fatima

enter number: 8

enter brith day:

day:26

month:3

year:1990


the data of employee is:

name    number    brithday.day    brithday.month    brithday.year

kamal     1           13               3                1984

fady      2           19               4                1995

banan     3           28               9                2007

yosof     5           82               10               2009

fatima    8           26               3                1990
```

****

الوحدة الثالثه

الملفات

Files

الملفات Files

- هذا الهيكل مهم جداً ، والبيانات ستخزن أولاً في وسائط التخزين الأولية ثم تخزن في وسائط التخزين الثانوية التي تكون البيانات موجودة فيها دائماً وإذا أردنا أن نجرى عملية معالجة على هذه البيانات تكون موجودة وإذا قطع التيار الكهربائي تكون موجودة وذلك مثل القرص الصلب (Hard disk).
هناك جزئية تقول : أن البيانات التي تخزن على وسائط التخزين الأولية إذا قطع التيار الكهربي (متطايرة) ، أما وسائط التخزين الثانوية تبقي البيانات الموجودة بها إذا قطع التيار الكهربي (غير متطايرة).
- عند إستخدام الملفات في البرنامج فإن البرنامج لا يتعامل مع وسائط التخزين الثانوية مباشرة وإنما هنالك خطوات ستكون موجودة وهي:-
  ١- سنقوم بتخزين البيانات على وسائط التخزين الأولية .
  ٢- ستقوم بنقل هذه البيانات وتخزينها على وسائط التخزين الثانوية .

وهذا يعني أن تخزين بيانات البرنامج تتم بطريقة غير مباشرة وكذلك استدعاء البيانات فإننا إذا أردنا أن نجرى عملية معالجة على البيانات يتم إرسال البيانات من الوسائط الثانوية الى الوسائط الأولية ثم عملية المعالجة .

تنقسم الملفات الى نوعين :

١- الملفات الداخلية (internal files) : بها ملف يسمي الملف المنطقي (Logical file) وتكون بياناته موجودة على الذاكرة الرئيسية وهي وسيط تخزين أولي أي أنه يفقد محتوياته إذا قطع التيار الكهربي .
٢- الملفات الخارجية (external files) : بها ملف يسمي الملف الفيزيائي (physical file) وتكون بياناته موجودة على وسائط التخزين الثانوية مثل القرص الصلب ويعني أنه لا يفقد محتوياته إذا قطع التيار الكهربي .

يوجد خازن وسيط مؤقت وهي ذاكرة الكيبورد (Buffer) وهي التي تتعامل مع الوسائط التخزينيه الثانوية .

- أولاً يتمالتخزين على الملف المنطقي وبعبارة ربط نقوم بنقل البيانات إلى الملف الفيزيائي ثم نتعرف على العبارات.
- الملف الفيزيائي يأخذ نفس أسماء الملفات العادية الموجوده على نظام التشغيل وهذا الملف يكون موجود على القرص الصلب.

للتعامل مع الملف المنطقي (Logical File) هنالك عدة خطوات -:

١- إنشاء الملف المنطقي : وتأتي من التصريح عن الملف المنطقي في لغة باسكال ومن الممكن أن أصرح عن طريقة إستخدام الكلمة المحجوزه type ولكن في الغالب يكون في منطقة التعريفات var .

```
var  logicalFileName : file of dataType;
```

Example:-

```
Var F : file of integer;
```

٢- هي عملية الربط بين الملف المنطقي والملف الفيزيائي (linked) ، والعبارة التي تستخدم للربط هي :

```
Assign(logicalFileName,'c:\physicalFileName.dat');
```

Example:-

```
Assign(f,'c:\student.dat');
```

c: : هي المنطقة التي نريد فيها حفظ الملف الفيزيائي على الذاكرة الثانوية ، وإذا كان الملف student غير موجود فإن هذه العبارة `'c:\student.dat'` ستمكننا من إنشاء الملف ، أما إذا كان الملف موجود فإنه سيقوم بعملية الربط مباشرة ، وعبارة assign تضمن لنا إنشاء الملف إذا لم يكن موجود.

٣- إستخدام الملف الفيزيائي :-

- يفتح الملف الفيزيائي إما بغرض الكتابة عليه أو بغرض القراءه منه أو الإضافه عليه.
- إذا فتح الملف (f) بغرض القراءة منه كذلك يفتح الملف (student) للقراءة منه بواسطة عملية الربط.
- للكتابة على الملف الفيزيائي يتم إستخدام العبارة التالية :

```
rewrite(logicalFileName);
rewrite(f);
```

هنا يفتح الملف الفيزيائي للكتابة عليه وكذلك الملف المنطقي ، وعبارة rewrite تعني أن ينتقل الكيرسل carsel إلى بداية الملف ، وكذلك عند فتح الملف الفيزيائي للكتابة علية يعني هذا مسح البيانات السابقة الموجوده عليه .

يمكن أن يكون هنالك ملف فيزيائي فتح وكتبت عليه بيانات ثم انتهى التعامل معه وبعد ذلك أراد المستخدم أن يتعامل مع هذا الملف مرة ثانية ليطبع البيانات الموجوده عليه على شاشة التنفيذ ، فإن ذلك يتم لهذه العبارة :

```
reset(logicalFileName);
reset(f);
```

وعندما نقوم بكتابة هذه العبارة لابد من معرفة هل أن الملف إنتهى أم لا ، وذلك ينتهي بواسطة الدالة end of file وتكتب كالتالي : `eof(logicalFileName)` .

سؤال: كيف نستطيع تخزين هذه البيانات على الملف المنطقي ، وبمعنى آخر كيف يمكن نقل البيانات من الملف المنطقي إلى الملف الفيزيائي؟؟؟
الجواب: يتم ذلك بإستخدام العبارة التالية :

```
write(logicalFileName , variableName)
```

variableName: هو المتغير المراد نقل قيمته إلى الملف الفيزيائي .

```
write(f,x)
```

- وللإضافة للملف الفيزيائي يتم إستخدام العبارة التالية :

```
append(logicalFileName);
append(f);
```

- وإذا أردنا أن نكسر الرابط بين الملف المنطقي والملف الفيزيائي يتم ذلك بهذه العبارة :

```
Cloce(logicalFileName);
Cloce(f);
```

Example:-

أكتب برنامج يعمل على تخزين 3 أعداد صحيحة على ملف فيزيائي يسمى integer.dat ثم يقوم البرنامج بطباعة بيانات الملف الفيزيائي على شاشة التنفيذ.

```
program tweleve;
var f:file of integer;
x,i:integer;
begin
assign(f,'c:\integer.dat');
rewrite(f);
for i:=1 to 3 do
begin
writeln('enter the value you longg wand add to phsical file:');
readln(x);
write(f,x);
end;
reset(f);
read(f,x);
write(x,' ');
while not eof(f) do
begin
read(f,x);
write(x,'  ');
end;
close(f);
readln;
end.
```

Output:-

```
enter the value you longg wand add to phsical file:
12
enter the value you longg wand add to phsical file:
4
enter the value you longg wand add to phsical file:
6
12 4  6
```

Example:-

أكتب برنامج يخزن عدد من الأعداد الموجبة على ملف فيزيائي إسمه real.dat ويتوقف البرنامج بإدخال القيمة (صفر) ومن ثم يقوم البرنامج بطابعة الملف على شاشة التنفيذ.

```
program therteen;
var f:file of real;
x:real;
begin
assign(f,'c:\real.dat');
rewrite(f);
writeln('enter x number:');
readln(x);
while (x<>0) do
if x>0 then
begin
write(f,x);
readln(x);
end
else writeln('enter number>0:');
readln(x);
reset(f);
read(f,x);
write(x,'  ');
while not eof(f) do
```

```
begin
read(f,x);
write(x,'  ');
end;
close(f);
readln;
end.
```

Output:-

```
enter x number:
12
4
6
54
33
3
0
 1.2000000000E+01   4.0000000000E+00   6.0000000000E+00   5.4000000000E+01
3.3
000000000E+01
```

تمرين(3):-

١- أكتب برنامج يعمل على تخزين سجلات 5 موظفين على ملف فيزيائي employee . dat ومن ثم يقوم البرنامج بطباعة سجلات الموظفين (بيانات الملف employee . dat) على شاشة التنفيذ علماً بأن بيانات الموظف هي (الإسم ، الرقم ، العنوان ، المرتب) .

٢- مستفيداً من البرنامج الأول قم بتخزين سجلات الموظفين في ملفين منفصلين (f1,f2)، الملف (employee1) يخزن سجلات الموظفين الذين تقل مرتباتهم عن ألف دينار والملف (employee2) يخزن بقية الملفات ثم قم بطباعة السجلات الثلاثة على شاشة التنفيذ.

حل التمرين:-

```
(1) program fourteen;
type
employee=record
name:string;
no:integer;
```

```
address:string;
salary:real;
end;
var f:file of employee;
i:integer;  emp:employee;
begin
assign(f,'d:\employee.dat');
rewrite(f);
writeln('enter the data of employee:');
for i:=1 to 5 do
with emp do
begin
write('enter name:',i);
readln(name);
write('enter number:');
readln(no);
write('enter address:');
readln(address);
write('enter salary:');
readln(salary);
write(f,emp);
end;
writeln('the data of employee is:');
writeln('name','    ','number','    ','address','    ','salary');
reset(f);
while not eof(f) do
begin
read(f,emp);
with emp do
writeln(name,'      ',no,'      ',address,'      ',salary);
end;
close(f);
readln;
end.
```

Output:-

```
enter the data of employee:
enter name:1 safiya
enter number:1
enter address:Iraq
enter salary:209.55
enter name:2 fatima
enter number:2
enter address:tunis
enter salary:133.554
enter name:3 haneen
enter number:3
enter address:turkya
enter salary:1233.554
enter name:4 banan
enter number:4
enter address:monester
enter salary:67776.444
enter name:5 alaa
enter number:5
enter address:sudan
enter salary:45456.45465
the data of employee is:
 name      number     address     salary
 safiya      1        Iraq        2.0955000000E+02
 fatima      2        tunis       1.3355400000E+02
 haneen      3        turkya      1.2335540000E+03
```

```
banan       4        monester    6.7776444000E+04
alaa        5        sudan       4.5456454650E+04
```

(2) program fiveteen;

```
type
employee=record
name:string;
no:integer;
address:string;
salary:real;
end;
var f:file of employee;
f1:file of employee;
f2:file of employee;
emp:employee; i:integer;
begin
assign(f,'d:\employee.dat');
assign(f1,'d:\employe1.dat');
assign(f2,'d:\employe2.dat');
reset(f);
rewrite(f1);
rewrite(f2);
while not eof(f) do
begin
read(f,emp);
if (emp.salary<1000) then
write(f1,emp)
else
write(f2,emp);
end;
reset(f1);
reset(f2);
writeln('the data of employee is:');
writeln('name','    ','number','    ','address','    ','salary');
reset(f);
```

```
while not eof(f) do
begin
read(f,emp);
with emp do
writeln(name,'      ',no,'      ',address,'      ',salary);
end;
writeln;
writeln('the data of employee whose salary less than 1000 D:');
writeln('name','    ','number','    ','address','    ','salary');
while not eof(f1) do
begin
read(f1,emp);
writeln(emp.name,'    ',emp.no,'          ',emp.address,'
',emp.salary);
end;
writeln;
writeln('the data of employee whose salary more than 1000 D:');
writeln('name','    ','number','    ','address','    ','salary');
while not eof(f2) do
begin
read(f2,emp);
writeln(emp.name,'    ',emp.no,'          ',emp.address,'
',emp.salary);
end;
close(f);
close(f1);
close(f2);
readln;
end.
```

Output:-

```
the data of employee is:
 name      number    address     salary
 safiya      1       Iraq        2.0955000000E+02
 fatima      2       tunis       1.3355400000E+02
 haneen      3       turkya      1.2335540000E+03
```

```
 banan        4       monester    6.7776444000E+04
 alaa         5       sudan       4.5456454650E+04
the data of employee whose salary less than 1000 D:
 name       number    address     salary
 safiya      1        Iraq        2.0955000000E+02
 fatima      2        tunis       1.3355400000E+02

the data of employee whose salary more than 1000 D:
 name       number    address     salary
 haneen      3        turkya      1.2335540000E+03
 banan       4        monester    6.7776444000E+04
 alaa        5        sudan       4.5456454650E+04
```

أما إذا كان مطلوب تخزين بيانات الموظفين على مصفوفة ومن ثم نقل بيانات المصفوفة وتخزينها على ملف فيزيائي يسمى emp.dat فإن الكود سيكون كالتالي :-

```
program new4;
type
employee=record
name:string;
no:integer;
address:string;
salary:real;
end;
var f:file of employee;
i:integer;
emp:array [1..10] of employee;
begin
assign(f,'d:\employee.dat');
rewrite(f);
writeln('enter the data of employee:');
for i:=1 to 5 do
with emp[i] do
```

```
begin
write('enter name: ');
readln(name);
write('enter number: ');
readln(no);
write('enter address: ');
readln(address);
write('enter salary: ');
readln(salary);
end;
for i:=1 to 10 do
write(f,emp[i]);
writeln('the data of employee is:');
writeln('name','    ','number','    ','address','    ','salary');
reset(f);
for i:=1 to 10 do
while not eof(f) do
begin
read(f,emp[i]);
with emp[i] do
writeln(name,'      ',no,'      ',address,'      ',salary);
end;
close(f);
readln;
end.
```

Output:-

```
enter the data of employee:
enter name: safiya
enter number: 3
enter address: khartoum
enter salary: 232.23434
enter name: yosof
enter number: 5
```

```
enter address: khartoum

enter salary: 454.545

enter name: najeh

enter number: 9

enter address: khartoum

enter salary: 324.121212

enter name: riad

enter number: 1

enter address: birmingham

enter salary: 3433.21223

enter name: khlood
enter number: 10

enter address: baghadad

enter salary: 45454.34343

the data of employee is:

name       number     address        salary

safiya       3        khartoum        2.3223434000E+02

yosof        5        khartoum        4.5454500000E+02

najeh        9        khartoum        3.2412121200E+02

riad         1        birmingham      3.4332122300E+03

khlood       10       baghadad        4.5454343430E+04
```

****

الوحدة الرابعه

مقدمة عن المؤشرات واللوائح المنغلقه

Overview to Pointers & Linked List

(4-1) مقدمة عن المؤشرات :-

سنتعرض هنا الى كيفية إستعمال لغة الباسكال في إنشاء بنية بيانات متحركه (dynamic data type) ، ونقصد بقولنا متحركة أي أنها تنمو (grow) أثناء تنفيذ البرنامج.

- Dynamic data Type : هي عبارة عن مجموعة من العناصر تسمي عقد nodes ، هذه العقد تربط مع بعضها بإستخدام المؤشرات ، وفي حقيقة الأمر أن العقد هي عبارة عن (record) سجلات منها ما يأخذ شكل خطي (عقده ثم عقده ويكون الربط في شكل لائحة) ، وبخلاف المصفوفات التي تشكل جزء لعدد محدود من العناصر ، فإن بنية البيانات المتحركة يمكن أن تمتد وتنكمش خلال تنفيذ البرنامج ويتوقف ذلك على حاجة البرنامج لتخزين البيانات .

بمعنى آخر : فإن عند الاعلان عن المصفوفة يتم حجر مساحة محدده من الذاكرة لا نستطيع تجاوزها عند استخدام المصفوفة ، وعند تجاوز هذه الساحة يحدث ما يسمي بمشكلة (array overflow) ، وهذه الحالة لا تحدث عند استخدام المؤشرات لأن الساحة المتاحة تكون مفتوحة.

إن بينة البيانات المتحركة تستخدم لتخزين بيانات دائمة التغيير ، مثال لذلك قائمة المسافرين على خطوط جوية معينة ، فإذا كان قد تم عمل هذه القائمة حسب الترتيب الهجائي لأسماء المسافرين وذلك بإستخدام مصفوفة ، فإنه عند حاجتنا لإضافة إسم مسافر إلى تلك القائمة فإنه يلزمنا أن نزيح جميع الأسماء التي تلي ذلك الإسم ، وذلك لإفساح المجال له ، لوضعه في مكانه الصحيح في المصفوفة ، ولا شك أن أن هذا الإجراء يتطلب عملاً شاقاً وزمناً طويلاً ، أما إذا إستخدمنا بنية البيانات المتحركة ، فإننا وبسهولة يمكننا إدخال ذلك الإسم بين إسمين من الأسماء في تلك القائمة لذلك توصف بنية البيانات المتحركة بأنها مرنة ، فمن السهل جداً إضافة بيانات جديدة عن طريق إنشاء عقده جديدة ، وإدخالها بين عقدتين موجودتين أصلاً ، ومن السهل إجراء أي تعديل على بنيه البيانات المتحركة ، كإجراء عملية حذف ، أو نقل لعقده ، وهذا أكثر كفاءه من إجراء تعديل على عنصر في مصفوفة ، حيث إن لكل عنصر مكان محدد مرتبط بالعناصر الأخرة يحدده مؤشره الدليلي .

الصيغة العامة:-

```
type
pointerName = ^dataType;
```

Example:-

```
type
p = ^node;
node = record
name : string;
no : integer;
place : string;
```

```
end;
var x,y,z:p;
```

. pointer operator : ^

. pointers : x,y,z

- الآن تم الإعلان عن المؤشر أو التصريح عنه ، ولكن كيف أستطيع أن أخصص له مساحة داخل ذاكرة الحاسوب حتى نستطيع نستخدم هذا المؤشر ؟؟؟؟
يتم ذلك بتخصيص مساحة للمؤشر بإستخدام الكلمة المحجوزة new .

Syntax:-

```
new (pointerVariable);
new (x); New (y); New (z);
```

إذاً المتغير x : هو عبارة عن سجل والسجل به ٣ حقول .

- بإستخدام المؤشرات أستطيع أن أصل إلى البيانات داخل ذاكرة الحاسوب سواء بتخزين البيانات أو إسترجاع بيانات.
- كذلك عندما نريد أن نصل إلى المواقع التي يشير إليها المؤشر يتم ذلك إما بتخزين بيانات عليه أو إسترجاع بيانات منه .
- يتم إستخدام عبارة read للكتابة على المؤشرات أو للتخزين .
- يتم إستخدام عبارة write للطباعة أو الإسترجاع .

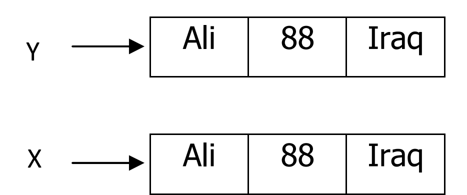
```
read/write(pointerVariable^.fieldName or variableName);
```

ومن الممكن أن يتم استخدام طريقة dot أو طريقة with وذلك كالتالي -:

```
with pointerVariable^ do
begin
filedName;
end;
```

- يمكننا أن نجعل مؤشر يشير إلى موقع يشير إليه مؤشر آخر بشرط أن تكون هذه المؤشرات من نفس النوع .

y:=x; هذا يعني أن y يشير إلى نفس الموقع الذي يشير إليه x ، وإذا قمت بطباعة y يطبع نفس بيانات x .



- إذا كنا في لحظة من اللحظات نريد أن نلغي تخصيص x هذا يعني أن نعمل deleting لعقدة معينة وبهذا ألغي التخصيص الذي يشير إلى العقده وذلك كالتالي :

```
dispose(pointerVariable);
dispose(x);
```

- أي عقدة شكلها أن يكون هنالك مؤشر يشير إليها ومؤشر خارج منها يشير إلى null حتى أستطيع أن أضيف إليها في أي وقت في نهايته .

head → **null**

Example:-

```
Type
P = ^node;
Node = record
Name : string;
No : integer;
Place : string;
End;
Var  x,y,z :p;
New(x);
New(y);
New(z);
{X^.name:='Ali';}
With x^ do
Begin
Name:='Ali';
```

```
No:=88;
Plase:'Iraq';
End;
Write(x^.name,' ',x^.no,' ',x^.place);
```

(4-2) اللوائح المنغلقه Linked List:-

هي عبارة عن سلسلة من العقد (nods) بحيث كل عقده مربوطة مع العقده التي تليها ، مؤشر العقد الأولي يسمي الرأس (head) والعقدة الأخيرة تثير الى null وبذلك نحصل على طرف لائحة خالي .

تستخدم السجلات في بناء اللوائح المنغلقه ، وأبسط سجل يتكون من حقلين حقل البيانات وحقل آخر مؤشر للسجل التالي .

تستخدم اللوائح المنغلقة على سبيل المثال في المترجمات ونظم التشغيل ونظم إدارة قواعد البيانات .

الإعلان عن عقدة مكونه من حقلين :-

```
program sixteen;
type
lptr=^node;
node=record
data:integer;
next:lptr;
end;
```

إنشاء اللائحة المنغلقة :-

الخوارزمية لإنشاء اللائحة المنغلقة:-

١- إنشاء مؤشر من النوع lptr .
٢- إجعل المؤشر يشير إلى null .

- ولإنشاء اللائحة المنغلقة نقوم بإستدعاء الدالة creatList في البرنامج الرئيسي فنكتب head:=creatList;.
- يمكن رسم شكل المؤشر في الذاكرة كالآتي : head ⟶ null .

```
function creatlist:lptr;
var p:lptr;
```

```
begin
new(p);
p^.next:=nil;
creatlist:=p;
end;
```

- ولإضافة أي عقدة إلى الائحة فإننا نقوم بإنشاء عقدة جديدة عن طريق الدالة getNode وهذه العقدة تكون من النوع node ، ونشير إليها بمؤشر من النوع lptr ويسمى newlptr ، ونقرأ قيمة الحقل data ونجعل المؤشر next يشير إلى null .

```
function getnode:lptr;
var newlptr:lptr;
begin
writeln('enter the integer you want add to the list:');
readln(n);
new(newlptr);
with newlptr^ do
begin
data:=n;
next:=nil;
end;
getnode:=newlptr;
end;
```

إضافة عقدة جديدة :-

خوارزمية الإضافة :-

١- إنشاء عقدة جديدة من النوع node تحتوي على العدد المراد إضافته إلى الائحة.
٢- إذا كان العدد في العقدة الجديدة أصغر من العدد في العقدة الأولى أضف العقدة الجديدة قبل العقدة الأولى .
٣- إذا كان العدد في العقدة الجديدة أكبر من العدد في العقدة الأولى أجر (٢) مع العقدة التالية مالم تصل إلى نهاية اللائحة المنغلقة.
٤- إذا وصلت إلى نهاية اللائحة أضف العقدة الجديدة في آخر اللائحة بإستخدام النداء الذاتي .

يمكن كتابة إجراء الإضافة كالتالي :-

```
procedure add(var head:lptr;newlptr:lptr);
var temp:lptr;
begin
if (head=nil) then
head:=newlptr
else
if (newlptr^.data=head^.data) then
writeln('this value is already in the list')
else
if (newlptr^.data<head^.data) then
begin
temp:=head;
head:=newlptr;
newlptr^.next:=temp;
end
else
if (newlptr^.data>head^.data) and (newlptr^.data<head^.next^.data)
then
begin
temp:=head^.next;
head^.next:=newlptr;
newlptr^.next:=temp;
end
else
add(head^.next,newlptr);
end;
```

- لحذف عقدة من اللائحة نقوم بعملية البحث عن تلك العقدة بالإجراء search التي ترجع مؤشرين أحدها يشير إلى العقدة المراد حذفها والتالي يشير إلى ماقبل العقدة المراد حذفها.

- الغرض لوجود مؤشرين هو جعل المؤشر قبل العقدة المراد حذفها يشير إلى نفس المكان الذي يشير إليه مؤشر العقدة المراد حذفها حتى يتسنى حذف العقدة المراد حذفها بالدالة dispose .

خوارزمية حذف عقدة من اللائحة المنغلقة :-

١- استخدم برنامج البحث في تحديد قيم المؤشرين target و pred .
٢- إذا كان المؤشر target يشير إلى null اطبع رسالى بعدم وجود العقدة المراد حذفها.
٣- إذا كان المؤشر target لا يشير إلى null بينما المؤشر pred يشير إلى null أجر مايلي (وهذا يعني أن العقدة المراد حذفها هي العقدة الأولى) :
أ- دع المؤشر head يشير إلى العقدة الثانية .
ب- إستخدم الدالة dispose لإلغاء العقدة الأولى .
٤- إذا كان المؤشران target و pred لا يشيران إلى null أجر مايلي:-
أ- اجعل pred^.next:=target^.next .
ب- استخدم الدالة dispose لإلغاء العقدة المشار إليها ب target .

```
procedure delete(var head:lptr; x:integer);
var target,pred:lptr;
begin
search(head,x,target,pred);
if (target=nil) then
writeln('this node is not found')
else
if (target=head) and (pred=nil) then
begin
head:=head^.next;
dispose(target);
end
else
if (target^.next=nil) then
begin
pred^.next:=nil;
```

```
dispose(target);
end
else
begin
pred^.next:=target^.next;
dispose(target);
end;
end;
```

البحث عن عقدة في اللائحة المنغلقة:-

خوارزمية البحث :-

١- إذا كانت اللائحة خالية من العقد أعطي القيمة null للمؤشرين target & pred .
٢- قارن بين الهدف والعقدة الأولى إذا كان الهدف يساوي العدد في العقدة الأولى دع المؤشر target يشير إليها واجعل المؤشر pred يشير إلى null .
٣- إذا كان الهدف لايساوي قيمة العدد في العقدة الأولى انتقل للمقارنة مع العقدة التالية ، إذا كان الهدف يساوي قيمة العدد في العقدة التالية دع المؤشر target يشير إليها ودع المؤشر pred يشير إلى العقدة التي قبلها .
٤- إذا كان الهدف لايساوي قيمة الحقل (العدد) في العقدة التالية كرر الخطوة (٣) .
٥- إذا وصلت إلى نهاية اللائحة المنغلقة ولم تجد الهدف أعطي قيمة null إلى المؤشرين target & pred .

```
procedure search (head:lptr; x:integer; var target,pred:lptr);
begin
if (head=nil) then
begin
writeln(x,'is not found');
target:=nil;
pred:=nil;
end
else
if (head^.data=x) then
begin
target:=head;
pred:=nil;
```

```
end
else
if (head^.next^.data=x) then
begin
target:=head^.next;
pred:=head;
end
else
if(x>head^.data) and (x<head^.next^.data) then
begin
writeln(x,'is not found');
target:=nil;
pred:=nil;
end
else
search(head^.next,x,target,pred);
end;
```

طباعة محتويات اللائحة المنغلقة :-

عند الحوجة لإسترجاع البيانات المخزنة نقوم بطباعة محتويات اللائحة المنغلقة.

خوارزمية طباعة محتويات اللائحة لمنغلقة :-

طالما أن المؤشر head لايساوي null أجر مايلي :-

أ- اطبع قيمة head^.data .

ب- اجعل head=head^.next .

```
procedure printlist(head:lptr);
begin
while head<>nil do
begin
writeln(head^.data,' ');
head:=head^.next;
end;
end;
```

مناداة البرامج السابقة داخل البرنامج الرئيسي :-

```
var n,no,r:integer;
head,x,y,b,s,target,pred:lptr;
begin {maim program}
new(head); new(x);
new(y); new(b);
head:=creatlist;
writeln('1--->creatlist','  ','2--->getnode','  ','3--->add to the
list',' ',
'4--->search from the list',' ','5--->print list',' ','6--->delete
node',' ',
'7--->exit');
repeat
writeln('what do you want???');
readln(no);
case no of
1:head:=creatlist;
2:s:=getnode;
3:add(head,s);
4:begin
writeln('enter number you want search:');
readln(r);
search(head,r,target,pred);
end;
5:printlist(head);
6:begin
writeln('enter no you want to dalete');
readln(r);
delete(head,r);
end;
end;
until (no=7);
if (no=7) then
writeln('you are out of the list');
writeln('by.......by I hope see you soon');
readln;     end.
```

output:-

```
1--->creatlist 2--->getnode 3--->add to the list 4--->search from
the list 5--->
print list 6--->delete node 7--->exit
what do you want???
1
what do you want???
2
enter the integer you want add to the list:
12
what do you want???
3
what do you want???
1
what do you want???
2
enter the integer you want add to the list:
8
what do you want???
3
what do you want???
12
8
what do you want???
4
enter number you want search:
8
what do you want???
4
enter number you want search:
100
100 is not found
what do you want???
6
enter no you want to delete
8
what do you want???
12
what do you want???
7
you are out of the list
by.......by I hope see you soon
```

Example:-

أكتب برنامج يقوم بتخزين سجلات ٥ موظفين ثم يقوم البرنامج بطباعة سجلات الموظفين على شاشة التنفيذ ويكون بإستخدام الرقم.

```
program seventeen;
type
lptr=^node;
node=record
name:string;
salary:real;
no:integer;
next:lptr;
end;
var head,d,l,m,target,pred:lptr;
    i,b,n,e:integer;
function creatlist:lptr;
var p:lptr;
begin
new(p);
p^.next:=nil;
creatlist:=p;
end;
function getnode:lptr;
var newlptr:lptr; r:integer; na:string; sa:real;
begin
writeln('enter employee name you want add to the list');
readln(na);
writeln('enter employee number you want add to the list');
readln(r);
writeln('enter employee salary you want add to the list');
readln(sa);
new(newlptr);
with newlptr^do
begin
name:=na;
no:=r;
salary:=sa;
```

```
next:=nil;
end;
getnode:=newlptr;
end;
procedure add(var head,newlptr:lptr);
var temp:lptr;
begin
if head=nil then
head:=newlptr
else
if newlptr^.no =head^.no then
writeln('this record alredy in the list')
else
if newlptr^.no<head^.no then
begin
temp:=head;
head:=newlptr;
newlptr^.next:=temp;
end
else
if(newlptr^.no>head^.no)and(newlptr^.no<head^.next^.no)then
begin
temp:=head^.next;
head^.next:=newlptr;
newlptr^.next:=temp;
end
else
add(head^.next,newlptr);
end;
procedure search(head:lptr; z:integer; var target,pred:lptr);
begin
if head=nil then
begin
writeln(z,'this number is not found in the list');
target:=nil;
pred:=nil;
```

```
end
else
if head^.no=z then
begin
target:=head;
pred:=nil;
end
else
if head^.next^.no=z then
begin
target:=head^.next;
pred:=head;
end
else
if(z>head^.no)and(z<head^.next^.no)then
begin
writeln('this record is not found in the list');
target:=nil;
pred:=nil;
end
else
search(head^.next,z,target,pred);
end;
procedure delete(var head:lptr; k:integer);
var target,pred:lptr;
begin
search(head,k,target,pred);
if target=nil then
writeln('this node is  not found in the list')
else
if(target=head)and(pred=nil)then
begin
head:=head^.next;
dispose(target);
end
else
```

```
if target^.next=nil then
begin
pred^.next:=nil;
dispose(target);
end
else
begin
pred^.next:=target^.next;
dispose(target);
end
end;
procedure printlist(head:lptr);
begin
writeln('name','    ','no','    ','salary');
while head<>nil do
begin
with head^ do
writeln(name,'   ',no,'   ',salary);
head:=head^.next;
end;
end;
begin{main program}
write('1--->creatlist',' ','2--->getnode && add',' ','3--->print
list',' ');
writeln('4--->search from the list',' ','5--->delete node',' ','6---
>exit');
repeat
writeln('what do you want???');
readln(b);
case b of
1:head:=creatlist;
2:begin
  for i:=1 to 3 do
  begin
  l:=getnode;
  add(head,l);
```

```
  end;
  end;
3:printlist(head);
4:begin
  writeln('enter the number of employee you want to search to the
record');
  readln(n);
  search(head,n,target,pred);
  if target<>nil then
  begin
  writeln('record is found in the listand the data of this employee
is:');
  with target^ do
  writeln('name is:',name,'no=',no,'salary=',salary);
  end
  end;
5:begin
  writeln('enter number of employee you want to delete from the
list');
  readln(e);
  delete(head,e);
  end;
  end;
until b=6;
if b=6 then
writeln('you are out of the list');
writeln('by......by');
readln;
end.
```

output:-

```
1--->creatlist 2--->getnode && add 3--->print list 4--->search from
the list 5--->delete node 6--->exit
what do you want???
1
what do you want???
```

```
2
enter employee name you want add to the list
safiya
enter employee number you want add to the list
1
enter employee salary you want add to the list
2323.445
enter employee name you want add to the list
najeh
enter employee number you want add to the list
2
enter employee salary you want add to the list
45454.7776
enter employee name you want add to the list
fatima
enter employee number you want add to the list
3
enter employee salary you want add to the list
43535.56546
what do you want???
3
name    no    salary
safiya   1    2.3234450000E+03
najeh    2    4.5454777600E+04
fatima   3    4.3535565460E+04
what do you want???
4
enter the number of employee you want to search to the record
2
record is found in the list and the data of this employee is:
name is:najeh  no=2  salary= 4.5454777600E+04
what do you want???
5
enter number of employee you want to delete from the list
3
what do you want???
```

```
3
name     no    salary
safiya  1     2.3234450000E+03
najeh   2     4.5454777600E+04
what do you want???
6
you are out of the list
by......by
```

أما إذا كان المطلوب ألا أراعي الترتيب سيتم تعديل procedure add وسيكون كالتالي :-

```
procedure add(var head,newlptr:lptr);
begin
if head=nil then
head:=newlptr
else
add(head^.next,newlptr);
end;
```

إن عملية النداء الذاتي دائما تتم بقيم جديدة فبالتالي عندما يتم مناداة procedure add مرة ثانية يكون ال head يشير إلى null .

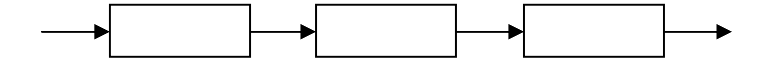
من أهم إستخدامات ال Linked List أو مميزاتها :-

١- الفهرسة الذاتيه : وتعني أنه إذا وصلنا إلى أي عقدة نستطيع بواسطتها أن نصل إلى بقية العقد المربوطة معها .
٢- إذا أردنا أن نضيف عقدة معينة لا نحاج أن نحجز أو نخصص مساحة لها وإنما فقط يتم التصريح عنها ، أي أنه لايوجد overflow أي المساحة غير محدودة وإنما المساحة تكون مفتوحة ، ومتى ما أردنا أن ننشئ عقدة نقوم بإنشائها.
٣- نستطيع إجراء كل العمليات المراد إجراءها على البيانات بسهولة أي أنها مرنة .

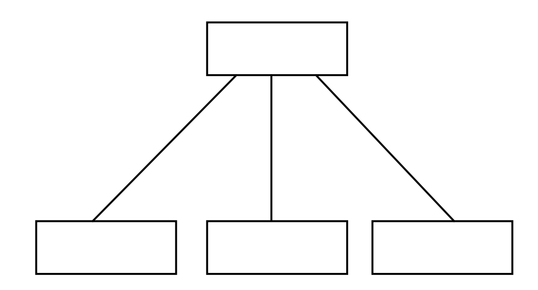
- إن هذا النوع من البيانات مرن جداً بحيث نجري العمليات على البيانات بسهولة تامة وهذا النوع من البيانات يستخدم في البيانات التي تتاج إلى تغيير مستمر .
- عندما ينتهي التعامل مع عقدة معينة من الممكن أن نقوم بمسح هذه العقدة ليتسنى لنا إستخدام مساحة هذه العقدة مرة ثانية.

- العقد ترتبط مع بعضها البعض بطريقتين هما :-

١- إما تكون خطية :



٢- إما أن تكون في شكل تفرع :



- في أي لائحة لابد أن يكون هنالك مؤشر يشير إليها.
- أي لائحة لا يوجد لديها مؤشر يشير إليها لا نستطيع إستخدامها أو الوصول إليها.
- أي عقدة لها مؤشران:-

١- مؤشر يشير إليها من خلاله نصل إلى كل حقول العقدة.

٢- مؤشر خارج منها يشير إلى null ولابد أن يكون المؤشران من نفس النوع .

****

الوحدة الخامسة

المكدسات

Stack

(١ – ٥) Stack Using Array :-

المكدسه هي قائمة من البيانات تحفظ بطريقة خطية ، يتم فيها حذف وإضافة البيانات من طرف واحد يسمى القمة Top أو stack Top (قمة المكدسه) ، وتكون هذه الطريقة خطية سواء أكانت dynamic or static .

هنالك ميزة يتصفبها الTop وهي :-

- أنه يحدد لنا الموقع لأخر عنصر تم تخزينه داخل المكدسه .
- عادة لا يحدد موقع فارغ وإنما يحدد الموقع لآخر عنصر.
- من خلاله نستطيع أن نعرف هل المكدسه خالية أو ممتلئه.

مبدأ المكدسه هي : First In First Out .

العمليات الأساسية للمكدس هي :-

١- عملية الإضافة Push .
٢- عملية الحذف Pop .

- مثلاً إذا أضفنا عنصر للمكدسه ، فإننا ترمز لعملية الإضافة ب push(s,a) وبعد إجراء هذه العملية فإن العنصر a يصبح في قمة المكدسه ، ولما كانت عملية الحذف لا تتم إلا من قمة المكدسه فقط فإنه يمكننا أن نرمز لها ب pop(s) ، ولا يوجد لزوم لتحديد العنصر المراد حذفه وذلك لأنه بعد إجراء هذه العملية يتم حذف العنصر الذي يقع في قمة المكدسه.

ماهي إستخدامات المكدسات :-

١- في النداء الذاتي .
٢- في البرامج الفرعية.
٣- في الإنتربط (عملية المقاطعة).
٤- في إيجاد التعابير الحسابية .

الإعلان عن المكدسه :-

نجد أن المكدسه يمكن الإعلان عنها كسجل يحتوي على حقلين :-

مصفوفة لإحتواء عناصر المكدسه ، ومتغير صحيح لتحديد الموقع الحالي لقمة المكدسه في المدى المحدد للمصفوفة ، ونقوم بتعريف متغير على أن نوعه من stack .

```
program seventeen;

const maxstack=100;

type

stack=record

item:array [1..maxstack] of integer;
top:0..maxstack;

end;

var s:stack;
```

لماذا تم تعريف المكدسه على أنه سجل ؟؟؟

حتى نستطيع الربط بين العناصر وبين المصفوفة .

- القمة top قيمتها تحدد موقع قمة المكدسه ، فمثلاً إذا كان top=3 ، فهذا يعني أن المكدسه تحتوي على ثلاثة عناصر item[1] , item[2] , item[3] ، وعندما تحدث عملية حذف pop من المكدسه ، فإن القمة top تتغير إلى ٢ ، أما إذا حدثت عملية إضافة push فإن القمة top سوف تزداد إلى ٤ ، و item[4] سوف يأخذ قيمة العنصر الجديد الذي تم إضافته .
- المكدسه الخالية لا تحتوي على أية عناصر ، وقد نستطيع الإستدلال على ذلك عندما تكون القمة top مساوية للصفر ، ولجعل المكدسه في حالتها الإبتدائية (مكدسة خالية) ، فإننا نجعل القمة مساوية للصفر top=0 .
- عنما تكون المكدسه ممتلئه نستطيع الإستدلال على ذلك عندما تكون القمة top مساوية للثابت maxStack .

الدالة empty :-

لإختبار ما إذا كانت المكدسه خالية أم لا ، فإننا نستخدم الدالة empty والتي ترجع القيمة true إذا كانت المكدسة خالية ، والقيمة false إذا كانت غير ذلك .

```
function empty(s:stack):boolean;
begin
if (s.top=0) then
empty:=true
else
empty:=false;
end;
```

الدالة full :-

وهي لإختبار هل المكدسه ممتلئه أم لا ، فترجع القيمة true إذا كانت المكدسه ممتلئه والقيمة false إذا كانت غير ذلك .

```
function full(s:stack):boolean;
begin
if (s.top=maxstack) then
full:=true
else
full:=false;
end;
```

خوارزمية الحذف من المكدسه :-

١- إختبار ما إذا كانت المكدسه خاليه أم لا ، فإذا كانت غير خالية استمر في تنفيذ بقية الخطوات ، وإلا فارجع إلى البرنامج المنادى .
٢- إحذف العنصر الذي يقع في قمة المكدسه .
٣- اطرح واحد من قمة المكدسه .
٤- إرجع العنصر المحذوف إلى البرنامج المنادى.

- لنفرض الآن أننا قمنا بإجراء عملية الحذف pop وتم تكرارها حتى آخر عنصر في المكدسه ، فإننا نحصل على مكدسة خالية emptyStack ، لذلك قبل إجراء عملية الحذف يجب التأكد من أن المكدسه ليست خالية ، وذلك بواسطة العملية empty(s) ، والتي تحدد ما إذا كانت المكدسه خالية أم لا .

```
function pop(var s:stack):integer;
begin
if empty(s) then
```

```
writeln('stack is empty')
else
begin
pop:=s.item[s.top];
s.top:=s.top-1;
end;
end;
```

- لنفرض الآن أن لدينا مكدسة خالية وقمنا بإجراء عملية الحذف pop فإننا نحصل على underFlow ، وبالتالي يجب فحص المكدسه بواسطة الدالة empty(s) قبل عملية الحذف.

خوارزمية الإضافه للمكدسه :-

١- إختبر ما إذا كانت المكدسه ممتلئه أم لا ، إذا لم تكن ممتلئه إستمرفي تنفيذ الخطوات وإلا فارجع إلى البرنامج المنادى .
٢- أضف واحد إلى قمة المكدسه .
٣- أسند العنصر المراد إضافته إلى قمة المكدسه.

- لنفترض أن لدينا مكدسة ممتلئه ، وقمنا بإجراء عملية إضافة push ، فإننا نحصل على overFlow ولذلك يمكن إستخدام الدالة full(s) قبل عملية الإضافة push لتفادي overFlow ، وهي ترجع true إذا كانت المكدسه ممتلئه ، كما ترجع القيمة المنطقية falseإذا كانت غير ذلك .

```
procedure push(var s:stack; x:integer);
begin
if full(s) then
writeln('stack is full')
else
begin
s.top:=s.top+1;
s.item[s.top]:=x;
end;
end;
```

عملية تحديد العنصر العلوي في المكدسه :-

```
function stacktop(s:stack):integer;
begin
```

```
if empty(s) then
writeln('stack is empty')
else
stacktop:=s.item[s.top];
end;
```

عملية التهيئه للمكدسه :-

```
procedure initialize(var s:stack);
begin
s.top:=0;
end;
```

عملية الإضافة (لجمع آخر عددين في المكدس) :-

```
procedure addtop(var s:stack);
var  op1,op2:integer;
element:boolean;
begin
op2:=pop(s);
if empty(s) then
begin
element:=true;
push(s,op2);
end
else
begin
op1:=pop(s);
element:=false;
push(s,op1+op2);
end;
end;
```

عملية addStack (هذا لتكرار الإجراء السابق وهو يكون لجمع جميع عناصر المكدسه):-

```
procedure addstack(var s:stack);
var element:boolean;
op1,op2:integer;
begin
repeat
addtop(s);
until element;
end;
```

عملية الطباعة:-

```
procedure display(s:stack);
var i:integer;
begin
if empty(s) then
writeln('stack is empty')
else
for i:=s.top downto 0 do
write(s.item[i],' ');
end;
```

مناداة جميع البرامج السابقة في البرنامج الرئيسي:-

```
var no,x,p,b:integer;
begin {main program}
write('1-->initialize','2-->empty','3-->full','4-->push','5-->pop','6--
>stacktop','10-->exite');
writeln('7-->display',' ','8-->addtop',' ','9-->addstack');
repeat
```

```
writeln('what do you want??');

readln(no);

case no of

1:initialize(s);

2:begin

if empty(s) then

writeln('the stack is empty')

else

writeln('the stack is not empty');

end;

3:begin

if full(s) then

writeln('the stack is full')

else

writeln('the stack is not empty');

end;

4:begin

writeln('enter x number:');

readln(x);

push(s,x);

end;

5:p:=pop(s);

6:b:=stacktop(s);

7:display(s);

8:addtop(s);

9:addstack(s);

end;
```

```
until (no=10);

write('end...');

readln;

end.
```

Output:-

```
1-->initialize 2-->empty 3-->full 4-->push 5-->pop 6-->stacktop 10-->exite
7-->display 8-->addtop 9-->addstack

what do you want??

1

what do you want??

2

the stack is empty

what do you want??

4

enter x number:

45

what do you want??

4

enter x number:

90

what do you want??

4

enter x number:

1

what do you want??

7

1 90 45

what do you want??

6
```

```
what do you want??
5
what do you want??
7
90 45
what do you want??
10
end...
```

-: Stack Using Pointer (٥ – 2)

- في هذا النوع نحافظ أيضاً على مبدأ ال stack وهو first in first out .
- مثلاً إذا كان المطلوب تخزين أعداد صحيحة ، فإننا هنا لا نحتاج إلى الدالة full(s) لأن المكدسه مساحتها مفتوحه لأنها daynamic data structure ، ولكن أحتاج إلى الدالة empty لأن مشكلة الـunderFlow لا زالت موجوده.

Example:-

```
program tewantyone;
type
stack=^node;
node=record
item:integer;
next:stack;
end;
var s:stack;
function empty(st:stack):boolean;
begin
if st=nil then
empty:=true
else
empty:=false;
end;

function pop(var s:stack):integer;
```

```
var p:stack;
begin
new(p);
if empty(s) then
writeln('stack is empty')
else
begin
pop:=s^.item;
p:=s;
s:=s^.next;
dispose(p);
end;
end;

procedure push(var s:stack; x:integer);
var p:stack;
begin
new(p);
writeln('enter x number:');
readln(x);
p^.item:=x;
p^.next:=s;
s:=p;
end;

procedure initialize(var s:stack);
begin
s:=nil;
end;

procedure display(s:stack);
begin
if empty(s) then
writeln('stack is empty')
else
while s<>nil do
```

```
begin
write(s^.item,' ');
s:=s^.next;
end;
end;

var no,x,p:integer;

begin{main program}
writeln('1-->initialize',' ','2-->push',' ','3-->pop',' ','4--
>display',' ','5-->exite');
repeat
writeln('what do you want???');
readln(no);
case no of
1:initialize(s);
2:push(s,x);
3:p:=pop(s);
4:display(s);
end;
until (no=5);
writeln('end...');
readln;
end.
```

Output:-

```
1-->initialize 2-->push 3-->pop 4-->display 5-->exite
what do you want???
1
what do you want???
2
enter x number:
12
what do you want???
2
enter x number:
```

```
7
what do you want???
2
enter x number:
67
what do you want???
4
67 7 12 what do you want???
3
what do you want???
4
7 12 what do you want???
5
end...
```

****

الوحدة السادسه

تحويل التعابير الحسابيه وإيجاد قيمها

تستخدم عدة طرق لتمثيل التعابير الحسابيه منها :-

١- الصيغة العادية (infix) والتي تستخدم الأقواس الدائريه مثل ((a+b-()() .
٢- الصيغة البعديه (postfix) ويطلق عليها أيضاً الصيغة البولنديه ، ويأتي المعامل الحسابي في هذه الصيغة عادة بعد المتغير ولا تستخدم هذه الطريقة الأقواس الدائرية مثل (+ab).
٣- الصيغة القبلية (prefix) ويسبق المعامل الحسابي في هذه الطريقة المتغير ، ولا تستخدم هذه الطريقة الأقواس أيضاً مثل (+ab) .

- سوف نستعرض ألية تحويل التعبير الحسابي من الصيغة العادية (infix) إلى صيغة (postfix) ، علماً بأن عملية التحويل إلى صيغة (prefix) تتم بطريقة مماثله .
- لإستخدام الـstack في إيجاد قيمة التعبير الحسابي لابد أولاً من تحويل هذا التعبير إلى صيغة (postfix) بإستخدام الـstack أو بعدما تحسب قيمة التعبير الممثل بصيغة (postfix) بإستخدام ال stack أيضاً .

التحويل إلى صيغة الـ(postfix) :-

يتم تحويل التعبير الحسابي من الصيغة العادية إلى صيغة (postfix) وذلك بإعتبار التعبير الحسابي سلسلة رمزية تقرأ رمزاً رمزاً ، وتستخدم الخوارزمية التالية لتنفيذ عمل التحويل :-

(١) مادام هنالك رموز في السلسة نفذ الخطوات التالية.
(٢) إقرأ الرمز من السلسلة .
(٣) (٣-١) إذا كان الرمز متغيراً احفظ هذا الرمز في ال stack.

(٣-٢) إذا كان الرمز قوساً دائرياً مفتوحاً احفظ هذا الرمز في الـstack .

(٣-٣) إذا كان الرمز قوساً دائرياً مغلقاً اسرجع كافة الرموز من الـstack وأضفها إلى صيغة الـ(postfix) حتى مصادفة القوس الدائري المفتوح ثم إلغي الأقواس .

(٣-٤) إذا كان الرمز معامل حسابي أضف هذا المعامل إلى الـstack بشريطة ألا تحتوي قيمة الـstack على معامل أولويته أعلى أو تساوي وإلا استرجع المعامل الأعلى أولية من الـstack والحقه إلى صيغة الـ(postfix) ثم احفظ المعامل المقروء في الـstack .

(٤) كرر الخطوات السابقه حتى نهاية السلسلة .

أما الأولويات للمعاملات هي :-

- إشارة الضرب والقسمة : الأولوية العليا .
- إشارة الجمع والطرح : الأولوية الدنيا .

مثال:-

إستخدم الخوارزمية السابقة لتحويل التعبير a+(b-c)*d من صيغة ال(infix) إلى صيغة ال(postfix) .

الحل:-

Symbole	Action	The output string after the action	The stack after this action
A	'a' is a variable we must put it to the output string.	a	empty
+	'+' is assign of operation we put it to the stack (stack is still empty so we needn't check the symbol's priority).	a	+
(	'(' is an opening bracket , we put it to the stack.	a	+(
B	'b' is a variable we must put it to the output string.	ab	+(
-	'-' is assign of operation it's priority is (2) now we must check the stack of the stack . It is priority (1). So we must put the current '-' symbol to the stack.	ab	+(-
C	'c' is a variable we must put it to the output string.	abc	+(-
)	')' is a closing bracket we must take out symbol from the stack till we find an opening bracket. Then we destroy both brackets.	abc-	+

*	'*' is assign of operation it's priority is (3) we must check the stack : There is a '+' symbol. On the top of the stack . It's priority is (2) and less than the priority of the '*' symbol. So we must put the current '*' symbol to the stack.	abc-	+*
d	'd' is a variable we must put it to the output string.	abc-d	+*

وعليه فإن صيغة الـ(postfix) المكافئة لصيغة الـ(infix) هي :-

abc-d*+

أما عملية إحتساب قيمة التعبير فتتم حسب الخوارزمية التالية :-

(١) مادام هنالك رموز في التعبير نفذ الخطوات التالية :-
(٢) إقرأ الرمز .
(٣) (٣-١) إذا كان الرمز قيمه احفظ هذه القيمه في ال stack.

(٣-٢) إذا كان الرمز معامل حسابي استرجع آخر موقعين في الـstack .

(٣-٣) إذا كان الرمز معامل اليساوي استرجع ماهو موجود في قمة الـstack .
(٤) كرر الخطوات السابقة .

فيما يلي :-

نستخدم الخوارزمية السابقة لإيجاد قيمة التعبير الحسابي التالي بإستخدام الـstack :

(752-4*+) .

الحل :-

Symbole	Action	The stack after this action
7	**'7' is a number. We put it to the stack.**	**7**
5	**'5' is a number. We put it to the stack.**	**75**
2	**'2' is a number. We put it to the stack.**	**752**
-	**'-' is assign of operation. We have to take out two numbers (5 and 2) from the stack and implement the (5-2=3) operation. Then we put the result to the stack.**	**73**
4	**'4' is a number. We put it to the stack.**	**734**
*****	**'*' is assign of operation. We have to take out two numbers (3 and 4) from the stack and implement the (3*4=12) operation. Then we put the result to the stack.**	**712**
+	**'+' is assign of operation. We have to take out two numbers (7 and 12) from the stack and implement the (7+12=19) operation. Then we put the result to the stack.**	**19**

يشير الشكل التالي إلى : كيفية إستخدام ال stack في إيجاد قيمة التعبير الحسابي الممثل بالصيغة البعديه (postfix) :-

210+96-1

Push 2

Push 10

pop 10

pop 2

Push 2+10=12

10

2

12

Push 9

Push 6

6

9

12

Pop 6

pop 9

Push 9-6=3



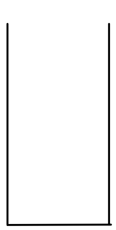
Pop 3

Push 12

Push 12/3=4



pop answer = 4



****

الوحدة السابعه

الصفوف

Queues

تعريف الصف :-

الصف هو قائمة من العناصر تحفظ بطريقة خطية بحيث يمكن حذف العناصر من طرف واحد يسمة المقدمة أو الرأس (front) بينما يتم إضافة العناصر من الطرف الآخر ، ويسمى المؤخرة أو الذيل (rear) والبيانات تكون في صورة خطية .

مبدأ الصف هو : القادم أولاً له الخدمة first in first out .

الصفوف في باسكال تنقسم إلى نوعين :-

١- الصفوف الخطية .
٢- الصفوف الدائرية : يمكن أن تكون daynamic أو static وسنتناول هنا الstatic .

- هناك ميزة أساسية تتسم بها مؤخرة الصف (rear) وهي أنها تحدد موقع فارغ ، أما المقدمة (front) فإنه يحدد موقع أول عنصر تم تخزينه في الqueue بواسطة الrear .
- الحالة الإبتدائية هي التي يكون بها الqueue فارغ وهو أن الrear وال front يكونان يحددان ال location رقم (٠) أي الحالة الإبتدائية ، وذلك يتم بمناداة الإجراء (initialize) وبذلك يكون الqueue مهيء حتى يستقبل العناصر (الحالة الإبتدائية تدل على أن rear و front يحددان الموقع (٠)).
- الqueue يكون فارغ إذا كان كل من الrear وfront يحددان لنفس الموقع بغض النظر عن ماإذا كان الموقع الذي يحددانه صفر أو غيره.
- أما إذا كنا نريد أن نختبر هل أن الصف ممتلئ أم لا ؟؟؟
  إذا كان الrear يقع خلف الfront مباشرة نقول أن الqueue ممتلئ .
- لابد أن نحافظ على أن الrear يحدد موقع فارغ ، فإذا تم تخزين عنصر فلابد أن يتم نقل الrear إلى نوقع فارغ.
- أما إذا كنا نريد أن نضيف عدد من الأعداد الصحيحة إلى الصف فإننا سنضيف العنصر مباشرة لأنه يتوفر لنا موقع فارغ ، أما عندما كنا في الstack كنا عندما ننادي push كان لابد لنا بعد ذلك أن نجعل ال top يصعد إلى موقع فارغ لأن ال top لا يحدد موقع فارغ .
- **العمليات الأساسية التي تجرى على الصف هي :-**
  ١- عملية تخزين العناصر insert .
  ٢- عملية الحذف delete .

بناء الصفوف بإستخدام المصفوفات في لغة باسكال

Queue Implementation Using Arrays in Pascal

الإعلان عن الصف :-

سنستخدم المصفوفة لتمثيل الصف في لغة باسكال ، وذلك بعد أن نضيف بعض الضوابط التي تحكم عملية الإضافة والحذف من الصف ، وسوف نستخدم متغيرين front و rear لتحديد العنصر الأول والعنصر الأخير في الصف ، بحيث يشير front إلى موقع العنصر الأول في الصف ، وrear إلى الموقع الذي يقع خلف موقع العنصر الأخير مباشرة في الصف ، أي أن موقع الrear سوف يكون خالياً دائماً وبالتالي فإن صف من الأعداد الصحيحة يمكن تمثيله بالآتي :-

```
program tewantythree;

const maxqueue=100;

type

queue=record

item:array[0..maxqueue] of integer;

rear,front:0..maxqueue;

end;

var q:queue;
```

- إن إستخدام المصفوفة لتمثيل الصف سوف يتيح الفرصة لحدوث overflow لأن المصفوفة محدودة العناصر .

الدالة (empty):-

- إن المتغير rear سوف يكون خالياً دوماً ، وهو يمثل مؤخرة الصف ، أي أن الصف يكون خالياً إذا كان front=rear وبذلك يمكن تمثيل العملية empty بالدالة التالية :-

```
function empty(q:queue):boolean;

begin
```

```
if (q.rear=q.front) then

empty:=true

else

empty:=false;

end;
```

الدالة (full):-

```
function full(q:queue):boolean;

begin

if q.front=(q.rear+1) mod maxqueue then

full:=true

else

full:=false;

end;
```

عملية الإضافة للصف (insert):-

```
procedure insert(var q:queue; x:integer);

begin

if full(q) then

writeln('queue is full')

else

begin

writeln('enter x number:');
```

```
readln(x);
q.item[q.rear]:=x;
q.rear:=(q.rear+1) mod maxqueue;
end;
end;
```

عملية الحذف من الصف (delet):-

```
function delet(var q:queue):integer;
begin
if empty(q) then
writeln('queue is empty')
else
begin
delet:=q.item[q.front];
q.front:=(q.front+1) mod maxqueue;
end;
end;
```

عملية التهيئه (initialize):-

```
procedure initialize(var q:queue);
begin
q.rear:=0;
q.front:=0;
```

```
end;
```

عملية الطباعة (display):-

```
procedure display(q:queue);
var i:integer;
begin
if (q.front>q.rear) then
for i:=q.front downto q.rear do
write(q.item[i],' ')
else
begin
for i:=q.front to q.rear do
write(q.item[i],' ');
end;
end;
```

مناداة البرامج الفرعية السابقة في البرنامج الرئيسي :-

```
var p,x,no,i:integer;
begin {main program}
writeln('1-->initialize',' ','2-->insert',' ','3-->delet',' ','4--
>display',' ','5-->exite');
repeat
writeln('what do you want???');
```

```
readln(no);

case no of

1:initialize(q);

2:begin

for i:=1 to 4 do

insert(q,x);

end;

3:p:=delet(q);

4:display(q);

end;

until (no=5);

writeln('end..');

readln;

end.
```

Output:-

```
1-->initialize 2-->insert 3-->delet 4-->display 5-->exite

what do you want???

1

what do you want???

2

enter x number:

34

enter x number:

4
```

```
enter x number:
2
what do you want???
4
34 4 2
what do you want???
3
what do you want???
4
4 2
what do you want???
5
end..
```

تمرين(٤):-

- **Linked List :-**

١- أكتب برنامج يقوم بترتيب الأعداد تصاعدياً مستخدماً اللوائح المنغلقة .

٢- اكتب برنامج يقوم بتخزين سجلات ١٠ موظفين مرتبة بإستخدام الإسم علماً بأن بيانات الموظف هي (الإسم ،الرقم،المرتب) ثم قم بطباعة هذه البيانات.

٣- أعد كتابة البرنامج السابق على ألا يكون هنالك ترتيب .

- **Stack Using Array :-**

١- أكتب برنامج يقوم بتخزين عدد من الأعداد الصحيحة داخل الstack ويتوقف البرنامج بإدخال عدد سالب أو صفر ، ثم يقوم البرنامج بطباعة هذه الأعداد على شاشة التنفيذ .

٢- مستخدماً الstack أكتب برنامج يقوم بتخزين ٣ أعداد صحيحة ومن ثم يقوم البرنامج بجمع هذه الأعداد الثلاثة وتخزين الناتج على الstack ثم طباعة ناتج الجمع على شاشة التنفيذ ومن ثم يقوم البرنامج بتحويل هذا الناتج من النظام العشري إلى النظام الثنائي وطباعة الرقم الثنائي على شاشة التنفيذ .

٣- أكتب برنامج يعمل على تخزين سجلات ٤ طلاب على أن يقوم بطباعة هذه السجلات على شاشة التنفيذ علماً بأن بيانات الطالب هي (الإسم،الرقم،العمر،العنوان).

- **Stack Using Pointer :-**

١- أكتب برنامج يخزن عدد من الأعداد الصحيحة ويتوقف البرنامج إذا كانت القيمة المدخلة فردية ، ثم يقوم البرنامج بطباعة هذه الأعداد على شاشة التنفيذ.
٢- أكتب برنامج يعمل على تخزين سجلات ٥ موظفين ثم يقوم البرنامج بطباعة بيانات الموظفين على شاشة التنفيذ علماً بأن بيانات الموظف هي (الإسم،الرقم،المرتب،العنوان).

- **Queue :-**

١- أكتب برنامج يعمل على تخزين سجلات ٣ طلاب ثم يقوم البرنامج بطباعة بيانات الطلاب على شاشة التنفيذ علماً بأن بيانات الطلاب هي (الإسم،الرقم،العمر).

حل التمرين:-

Linked List:-

(1)
```
program sixteen;

type

lptr=^node;

node=record

data:integer;

next:lptr;

end;

var n,no,r:integer;

head,x,y,b,s,target,pred:lptr;


function creatlist:lptr;

var p:lptr;

begin

new(p);

p^.next:=nil;
```

```
creatlist:=p;
end;
function getnode:lptr;
var newlptr:lptr;
begin
writeln('enter the integer you want add to the list:');
readln(n);
new(newlptr);
with newlptr^ do
begin
data:=n;
next:=nil;
end;
getnode:=newlptr;
end;


procedure add(var head:lptr;newlptr:lptr);
var temp:lptr;
begin
if (head=nil) then
head:=newlptr
else
if (newlptr^.data=head^.data) then
```

```
writeln('this value is already in the list')
else
if (newlptr^.data<head^.data) then
begin
temp:=head;
head:=newlptr;
newlptr^.next:=temp;
end
else
if (newlptr^.data>head^.data) and (newlptr^.data<head^.next^.data) then
begin
temp:=head^.next;
head^.next:=newlptr;
newlptr^.next:=temp;
end
else
add(head^.next,newlptr);
end;

procedure search (head:lptr; x:integer; var target,pred:lptr);
begin
if (head=nil) then
begin
```

```
writeln(x,'is not found');
target:=nil;
pred:=nil;
end
else
if (head^.data=x) then
begin
target:=head;
pred:=nil;
end
else
if (head^.next^.data=x) then
begin
target:=head^.next;
pred:=head;
end
else
if(x>head^.data) and (x<head^.next^.data) then
begin
writeln(x,'is not found');
target:=nil;
pred:=nil;
end
```

```
else

search(head^.next,x,target,pred);

end;


procedure delete(var head:lptr; x:integer);

var target,pred:lptr;

begin

search(head,x,target,pred);

if (target=nil) then

writeln('this node is not found')

else

if (target=head) and (pred=nil) then

begin

head:=head^.next;

dispose(target);

end

else

if (target^.next=nil) then

begin

pred^.next:=nil;

dispose(target);

end

else
```

```
begin
pred^.next:=target^.next;
dispose(target);
end;
end;


procedure printlist(head:lptr);
begin
while head<>nil do
begin
writeln(head^.data,' ');
head:=head^.next;
end;
end;
begin {maim program}
new(head); new(x);
new(y); new(b);
head:=creatlist;
writeln('1--->creatlist',' ','2--->getnode',' ','3--->add to the list','
',
'4--->search from the list',' ','5--->print list',' ','6--->delete node','
',
'7--->exit');
repeat
```

```
writeln('what do you want???');

readln(no);

case no of

1:head:=creatlist;

2:s:=getnode;

3:add(head,s);

4:begin

writeln('enter number you want search:');

readln(r);

search(head,r,target,pred);

end;

5:printlist(head);

6:begin

writeln('enter no you want to dalete');

readln(r);

delete(head,r);

end;

end;

until (no=7);

if (no=7) then

writeln('you are out of the list');

writeln('by.......by I hope see you soon');

readln;
```

```
end.
```

Output:-

```
1--->creatlist 2--->getnode 3--->add to the list 4--->search from the list 5--->
print list 6--->delete node 7--->exit
what do you want???
1
what do you want???
2
enter the integer you want add to the list:
54
what do you want???
3
what do you want???
2
enter the integer you want add to the list:
34
what do you want???
3
what do you want???
5
34
54
what do you want???
4
enter number you want search:
433
433 is not found
what do you want???
```

```
6
enter no you want to dalete
34
what do you want???
5
54
what do you want???
7
you are out of the list
by.......by I hope see you soon
```

(2) program seventeen;

```
type
lptr=^node;
node=record
name:string;
salary:real;
no:integer;
next:lptr;
end;
var head,x,y,z,target,pred:lptr;
i,b:integer;
s,c:string;

function creatlist:lptr;
var p:lptr;
begin
new(p);
p^.next:=nil;
```

```
creatlist:=p;
end;

function getnode:lptr;
var newlptr:lptr; number:integer; nam:string; sal:real;
begin
writeln('enter employee name: ');
readln(nam);
writeln('enter employee number: ');
readln(number);
writeln('enter employee salary: ');
readln(sal);
new(newlptr);
with newlptr^do
begin
name:=nam;
no:=number;
salary:=sal;
next:=nil;
end;
getnode:=newlptr;
end;

procedure add(var head,newlptr:lptr);
var temp:lptr;
begin
if head=nil then
head:=newlptr
else
```

```
if newlptr^.name =head^.name then
writeln('this record alredy in the list')
else
if newlptr^.name<head^.name then
begin
temp:=head;
head:=newlptr;
newlptr^.next:=temp;
end
else
if(newlptr^.name>head^.name)and(newlptr^.name<head^.next^.name)then
begin
temp:=head^.next;
head^.next:=newlptr;
newlptr^.next:=temp;
end
else
add(head^.next,newlptr);
end;

procedure search(head:lptr; w:string; var target,pred:lptr);
begin
if head=nil then
begin
writeln(w,'this name is not found in the list');
target:=nil;
pred:=nil;
end
else
```

```
if head^.name=w then
begin
target:=head;
pred:=nil;
end
else
if head^.next^.name=w then
begin
target:=head^.next;
pred:=head;
end
else
if(w>head^.name)and(w<head^.next^.name)then
begin
writeln('this record is not found');
target:=nil;
pred:=nil;
end
else
search(head^.next,w,target,pred);
end;

procedure delete(var head:lptr; q:string);
var target,pred:lptr;
begin
search(head,q,target,pred);
if target=nil then
writeln('this node is  not found')
else
```

```
if(target=head)and(pred=nil)then
begin
head:=head^.next;
dispose(target);
end
else
if target^.next=nil then
begin
pred^.next:=nil;
dispose(target);
end
else
begin
pred^.next:=target^.next;
dispose(target);
end
end;

procedure printlist(head:lptr);
begin
writeln('name','    ','no','    ','salary');
while head<>nil do
begin
with head^ do
writeln(name,'   ',no,'   ',salary);
head:=head^.next;
end;
end;
```

```
begin{main program}
write('1--->creatlist',' ','2--->getnode && add',' ','3--->print',' ');
writeln('4--->search',' ','5--->delete',' ','6--->exit');
repeat
writeln('what do you want???');
readln(b);
case b of
1:head:=creatlist;
2:begin
  for i:=1 to 3 do
  begin
  y:=getnode;
  add(head,y);
  end;
  end;
3:printlist(head);
4:begin
  writeln('enter the name of employee you want to search: ');
  readln(s);
  search(head,s,target,pred);
  if target<>nil then
  begin
  writeln('record is found in the listand the data is: ');
  with target^ do
  writeln('name is:',name,'no=',no,'salary=',salary);
  end
  end;
5:begin
  writeln('enter name of employee you want to delete: ');
```

```
  readln(c);
  delete(head,c);
  end;
  end;
until b=6;
if b=6 then
writeln('end.....');
readln;
end.
```

Output:-

```
1--->creatlist 2--->getnode && add 3--->print 4--->search 5--->delete 6---
>exit
what do you want???
1
what do you want???
2
enter employee name:
safiya
enter employee number:
1
enter employee salary:
1323.3434
enter employee name:
najeh
enter employee number:
2
enter employee salary:
2323.5565
enter employee name:
```

```
noori
enter employee number:
3
enter employee salary:
3434.3545
what do you want???
3
name    no    salary
najeh   2     2.3235565000E+03
noori   3     3.4343545000E+03
safiya  1     1.3233434000E+03
what do you want???
5
enter name of employee you want to delete:
noori
what do you want???
3
name    no    salary
najeh   2     2.3235565000E+03
safiya  1     1.3233434000E+03
what do you want???
6
end.....
```

(٣) إذا لايكون هنالك ترتيب سيكون هذا البرنامج هو نفس البرنامج السابق مع إختلاف الإجراء add وسيكون كالتالي :-

```
procedure add(var head,newlptr:lptr);
begin
if head = nil then
```

```
head:=newlptr
else
add(head^.next,newlptr);
end;
```

Stack Using Array:-

(1) `program eighteen;`

```
const maxstack=100;
type
stack=record
item:array [1..maxstack] of integer;
top:0..maxstack;
end;
var s:stack;

function empty(s:stack):boolean;
begin
if (s.top=0) then
empty:=true
else
empty:=false;
end;
```

```
function full(s:stack):boolean;

begin

if (s.top=maxstack) then

full:=true

else

full:=false;

end;


function pop(var s:stack):integer;

begin

if empty(s) then

writeln('stack is empty')

else

begin

pop:=s.item[s.top];

s.top:=s.top-1;

end;

end;


procedure push(var s:stack; x:integer);

begin

if full(s) then

writeln('stack is full')
```

```
else

while (x>0) do

begin

s.top:=s.top+1;

s.item[s.top]:=x;

writeln('enter x number:');

readln(x)

end;

end;


procedure initialize(var s:stack);

begin

s.top:=0;

end;


function stacktop(s:stack):integer;

begin

if empty(s) then

writeln('stack is empty')

else

stacktop:=s.item[s.top];

end;
```

```
procedure display(s:stack);

var i:integer;

begin

if empty(s) then

writeln('stack is empty')

else

begin

for i:=s.top downto 0 do

write(s.item[i],' ');

end;

end;


procedure addtop(var s:stack);

var  op1,op2:integer;

element:boolean;

begin

op2:=pop(s);

if empty(s) then

begin

element:=true;

push(s,op2);

end

else
```

```
begin
op1:=pop(s);
element:=false;
push(s,op1+op2);
end;
end;

procedure addstack(var s:stack);
var element:boolean;
op1,op2:integer;
begin
repeat
addtop(s);
until element;
end;

var no,x,i,p,b:integer;
begin {main program}
write('1-->initialize','2-->empty','3-->full','4-->push','5-->pop','6--
>stacktop','10-->exite');
writeln('7-->display',' '{,'8-->addtop',' ','9-->addstack'});
repeat
writeln('what do you want??');
readln(no);
```

```
case no of

1:initialize(s);

2:begin

if empty(s) then

writeln('the stack is empty')

else

writeln('the stack is not empty');

end;

3:begin

if full(s) then

writeln('the stack is full')

else

writeln('the stack is not empty');

end;

4:begin

writeln('enter x number');

readln(x);

push(s,x);

end;

5:p:=pop(s);

6:b:=stacktop(s);

7:display(s);

8:addtop(s);
```

```
9:addstack(s);

end;

until (no=10);

writeln('end.....');

readln;

end.
```

Output:-

```
1-->initialize2-->empty3-->full4-->push5-->pop6-->stacktop10-->exite7--
>display


what do you want??

1

what do you want??

2

the stack is empty

what do you want??

4

enter x number

12

enter x number:

11

enter x number:

3

enter x number:

444

enter x number:

2
```

```
enter x number:
0
what do you want??
7
2 444 3 11 12 ،
what do you want??
2
the stack is not empty
what do you want??
5
what do you want??
7
444 3 11 12 ،
what do you want??
10
end...
```

(2)

```
program nineteen;
const maxstack=100;
type
stack=record
item:array [1..maxstack] of integer;
top:0..maxstack;
end;
var s:stack;

function empty(s:stack):boolean;
begin
if (s.top=0) then
```

```
empty:=true
else
empty:=false;
end;

function full(s:stack):boolean;
begin
if (s.top=maxstack) then
full:=true
else
full:=false;
end;

function pop(var s:stack):integer;
begin
if empty(s) then
writeln('stack is empty')
else
begin
pop:=s.item[s.top];
s.top:=s.top-1;
end;
end;

procedure push(var s:stack; x:integer);
var i:integer;
begin
if full(s) then
writeln('stack is full')
```

```
else
begin
s.top:=s.top+1;
s.item[s.top]:=x;
end;
end;

procedure initialize(var s:stack);
begin
s.top:=0;
end;

function stacktop(s:stack):integer;
begin
if empty(s) then
writeln('stack is empty')
else
stacktop:=s.item[s.top];
end;

procedure display(s:stack);
var i:integer;
begin
if empty(s) then
writeln('stack is empty')
else
begin
for i:=s.top downto 0 do
write(s.item[i],' ');
```

```
end;
end;

procedure addtop(var s:stack);
var  op1,op2:integer;
element:boolean;
begin
op2:=pop(s);
if empty(s) then
begin
element:=true;
push(s,op2);
end
else
begin
op1:=pop(s);
element:=false;
push(s,op1+op2);
end;
end;

procedure addstack(var s:stack);
var element:boolean;
op1,op2:integer;
begin
repeat
addtop(s);
until element;
end;
```

```
var f1,f2:boolean;

no,x,i,p,b,op1,op2:integer;

begin {main program}

write('1-->initialize','2-->empty','3-->full','4-->push','5-->pop','6--
>stacktop','10-->exite');

writeln('7-->display',' ','8-->addtop',' ','9-->addstack');

repeat

writeln('what do you want??');

readln(no);

case no of

1:initialize(s);

2:begin

if empty(s) then

writeln('the stack is empty')

else

writeln('the stack is not empty');

end;

3:begin

if full(s) then

writeln('the stack is full')

else

writeln('the stack is not empty');

end;

4:begin

for i:=1 to 3 do

begin

writeln('enter x number:');

readln(x);

push(s,x);
```

```
end;
end;
5:p:=pop(s);
6:begin
b:=stacktop(s);
writeln('sum=',b);
x:=b mod 2;
writeln('decimal number=',x);
end;
7:display(s);
8:addtop(s);
9:addstack(s);
end;
until (no=10);
writeln('end...');
readln;
end.
```

Output:-

```
1-->initialize2-->empty3-->full4-->push5-->pop6-->stacktop10-->exite7--
>display
8-->addtop 9-->addstack
what do you want??
1
what do you want??
4
enter x number:
12
enter x number:
```

```
1
enter x number:
44
what do you want??
7
44 1 12
what do you want??
8
what do you want??
9
what do you want??
6
sum=57
decimal number=1
what do you want??
10
end...
```

(3) program tewanty;

```
const maxstack=5;
type
studant=record
name:string;
no:integer;
age:integer;
address:string;
end;
```

```
stack=record

item:array[1..maxstack] of studant;

top:0..maxstack;

end;

var s:stack;


function empty(s:stack):boolean;

begin

if (s.top=0) then

empty:=true

else

empty:=false;

end;


function full(s:stack):boolean;

begin

if (s.top=maxstack) then

full:=true

else

full:=false;

end;


procedure push(var s:stack; x:integer);
```

```
begin

if full(s) then

writeln('stack is full')

else

s.top:=s.top+1;

end;


procedure initialize(var s:stack);

begin

s.top:=0;

end;


procedure display(s:stack);

var i:integer;

begin

if empty(s) then

writeln('stack is empty')

else

begin

writeln('name','    ','number','    ','age','    ','address');

for i:=s.top downto 1 do

writeln(s.item[i].name,'    ',s.item[i].no,'    ',s.item[i].age,'
',s.item[i].address);

end;
```

```
end;


var name:string; address:string;

n,x,i,age,no:integer;

begin {main program}

writeln('1-->initialize',' ','2-->empty',' ','3-->full',' ','4-->push','
','5-->display',' ','6-->exite');

repeat

writeln('what do you want??');

readln(n);

case n of

1:initialize(s);

2:begin

if empty(s) then

writeln('the stack is empty')

else

writeln('the stack is not empty');

end;

3:begin

if full(s) then

writeln('the stack is full')

else

writeln('the stack is not empty');

end;
```

```
4:begin

for i:=1 to 4 do

begin

write('enter studant name:',i,' ');

readln(s.item[i].name);

write('enter studant number:',i,' ');

readln(s.item[i].no);

write('enter studant age:',i,' ');

readln(s.item[i].age);

write('enter studant address:',i,' ');

readln(s.item[i].address);

push(s,x);

end;

end;

5:display(s);

end;

until (n=6);

writeln('end...');

readln;

end.
```

Output:-

```
1-->initialize 2-->empty 3-->full 4-->push 5-->display 6-->exite

what do you want??

1
```

```
what do you want??
4
enter studant name:1 safiya
enter studant number:1 1
enter studant age:1 19
enter studant address:1 Iraq
enter studant name:2 banan
enter studant number:2 2
enter studant age:2 5
enter studant address:2 tunis
enter studant name:3 haneen
enter studant number:3 3
enter studant age:3 25
enter studant address:3 monester
enter studant name:4 fatima
enter studant number:4 4
enter studant age:4 21
enter studant address:4 khartoum
what do you want??
5
name      number    age     address
fatima     4        21      khartoum
haneen     3        25      monester
banan      2        5       tunis
safiya     1        19      Iraq
what do you want??
6
end...
```

Stack Using Pointer:-

```
(1)program tewantytwo;
type
stack=^node;
node=record
item:integer;
next:stack;
end;
var s:stack;
function empty(s:stack):boolean;
begin
if s=nil then
empty:=true
else
empty:=false;
end;


function pop(var s:stack):integer;
var p:stack;
begin
new(p);
if empty(s) then
writeln('stack is empty')
```

```
else
begin
pop:=s^.item;
p:=s;
s:=s^.next;
dispose(p);
end;
end;


procedure push(var s:stack; x:integer);
var p:stack;
begin
writeln('enter x number:');
readln(x);
while x mod 2=0 do
begin
p^.item:=x;
p^.next:=s;
s:=p;
new(p);
writeln('enter x number:');
readln(x);
end;
```

```
end;


procedure initialize(var s:stack);

begin

s:=nil;

end;


procedure display(s:stack);

begin

if empty(s) then

writeln('stack is empty')

else

while s<>nil do

begin

write(s^.item,' ');

s:=s^.next;

end;

end;


var no,x,p:integer;


begin{main program}

writeln('1-->initialize',' ','2-->push',' ','3-->display',' ','4-->pop','
','5-->exite');
```

```
repeat

writeln('what do you want???');

readln(no);

case no of

1:initialize(s);

2:push(s,x);

3:display(s);

4:p:=pop(s);

end;

until (no=5);

readln;

end.
```

Output:-

```
1-->initialize 2-->push 3-->display 4-->pop 5-->exite

what do you want???

1

what do you want???

2

enter x number:

12

enter x number:

4

enter x number:

5

what do you want???
```

```
3
4 12
what do you want???
5
```

(2)`program tewantufive;`

```
type
stack=^node;
node=record
name:string;
address:string;
no:integer;
salary:real;
next:stack;
end;
var s:stack;

function empty(s:stack):boolean;
begin
if s=nil then
empty:=true
else
empty:=false;
end;
```

```
function pop(var s:stack):integer;

var p:stack;

begin

new(p);

if empty(s) then

writeln('stack is empty')

else

begin

pop:=s^.no;

p:=s;

s:=s^.next;

dispose(p);

end;

end;


procedure push(var s:stack);

var p:stack; na,addr:string;

r,i:integer; sa:real;

begin

for i:=1 to 2 do

begin

write('enter name:',i,' ');
```

```
readln(na);

write('enter address:',i,' ');

readln(addr);

write('enter number:',i,' ');

readln(r);

write('enter salary:',i,' ');

readln(sa);

new(p);

p^.name:=na;

p^.address:=addr;

p^.no:=r;

p^.salary:=sa;

p^.next:=s;

s:=p;

end;

end;


procedure initialize(var s:stack);

begin

s:=nil;

end;


procedure display(s:stack);
```

```
begin

if empty(s) then

writeln('stack is empty')

else

writeln('name','    ','address','    ','number','    ','salsry');

while s<>nil do

begin

writeln(s^.name,'  ',s^.address,'  ',s^.no,'  ',s^.salary);

s:=s^.next;

end;

end;


var no,x,p:integer;

begin{main program}

writeln('1-->initialize',' ','2-->push',' ','3-->pop',' ','4-->display','
','5-->exite');

repeat

writeln('what do you want???');

readln(no);

case no of

1:initialize(s);

2:push(s);

3:p:=pop(s);

4:display(s);
```

```
end;

until (no=5);

writeln('end...');

readln;

end.
```

Output:-

```
1-->initialize 2-->push 3-->pop 4-->display 5-->exite
what do you want???
1
what do you want???
2
enter name:1 yosof
enter address:1 khartoum
enter number:1 4
enter salary:1 123.334
enter name:2 banan
enter address:2 khartoum
enter number:2 7
enter salary:2 23324.3332
enter name:3 haneen
enter address:3 khartoum
enter number:3 1
enter salary:3 2334.5654
enter name:4 rashed
enter address:4 khartoum
enter number:4 9
enter salary:4 232.5654
```

```
enter name:5 riad
enter address:5 birmingham
enter number:5 12
enter salary:5 2334232.1121
what do you want???
4
name      address      number    salsry
riad      birmingham   12        2.3342321121E+06
rashed    khartoum     9         2.3256540000E+02
haneen    khartoum     1         2.3345654000E+03
banan     khartoum     7         2.3324333200E+04
yosof     khartoum     4         1.2333400000E+02
what do you want???
5
end...
```

Queue:-

```
program tewantysix;
const maxqueue=10;
type
studant=record
name:string;
number:integer;
age:integer;
end;
queue=record
```

```
item:array[0..maxqueue] of studant;

rear,front:0..maxqueue;

end;

var q:queue;


function empty(q:queue):boolean;

begin

if (q.rear=q.front) then

empty:=true

else

empty:=false;

end;


function full(q:queue):boolean;

begin

if q.front=q.rear+1 mod maxqueue then

full:=true

else

full:=false;

end;


procedure insert(var q:queue);

begin
```

```
if full(q) then

writeln('queue is full')

else

q.rear:=(q.rear+1) mod maxqueue;

end;


procedure initialize(var q:queue);

begin

q.rear:=0;

q.front:=0;

end;


procedure display(q:queue);

var i:integer;

begin

writeln('name','    ','number','    ','age');

if (q.front>q.rear) then

for i:=q.front downto q.rear do

writeln(q.item[i].name,'    ',q.item[i].number,'    ',q.item[i].age)

else

begin

for i:=q.front to q.rear do

writeln(q.item[i].name,'    ',q.item[i].number,'    ',q.item[i].age)
```

```
end;

end;


var p,x,no,i,number,age:integer;

name:string;

begin {main program}

writeln('1-->initialize',' ','2-->insert',' ','3-->display',' ','4--
>exite');

repeat

writeln('what do you want???');

readln(no);

case no of

1:initialize(q);

2:begin

for i:=1 to 3 do

begin

write('enter studant name:',i,' ');

readln(q.item[i].name);

write('enter studant number:',i,' ');

readln(q.item[i].number);

write('enter studant age:',i,' ');

readln(q.item[i].age);

insert(q);

end;
```

```
end;

3:display(q);

end;

until (no=4);

writeln('end..');

readln;

end.
```

Output:-

```
1-->initialize 2-->insert 3-->display 4-->exite
what do you want???
1
what do you want???
2
enter studant name:1 alaa
enter studant number:1 5
enter studant age:1 10
enter studant name:2 khaked
enter studant number:2 8
enter studant age:2 11
enter studant name:3 banan
enter studant number:3 4
enter studant age:3 8
what do you want???
3
name    number    age
alaa     5         10
```

```
khaked    8        11

banan     4        8

what do you want???

4

end..
```

****

الوحدة الثامنه

خوارزميات البحث

Searching Algorithms

(١-٨) خوارزمية البحث الخطي Linear Search :-

- البحث الخطي يبحث في بيانات غير مرئية .
- البحث في مصفوفة مثلاً عن عنصر " صدف " إذا وجدنا العنصر نحدد موقعه وإلا نعطي رسالة بعدم وجوده .

الخوارزمية :-

١- أصف العنصر المراد البحث عنه "item " إلى نهاية المصفوفة A(n+1)s = item .

٢- أجعل Loc =1 .

٣- كرر أ وب بحيث A(loc) # item .

أ- Loc = Loc+1 .

ب- انهي التكرار.

٤- إذا كان Loc = N+1 أظهر رسالة بعدم وجود العنصر والإ اكتب موقعه وهو الـ Loc .

مثال (١) :- إذا كانت لدينا المصفوفة التالية

	1	2	3
A	١٣	٣	١٨

ابحث عن الرقم (٣)

N=3 ⟶ N+1=4

	1	2	3	4
A	13	3	18	3

Loc=1

A[Loc]#item ⟶ A[1]#3 ⟶ 13#3 ⟶ T.

Loc=2

A[Loc]#item ⟶ A[2]#3 ⟶ 3#3 ⟶ F.

إذاً الرقم (٣) موجود في الموقع الثاني.

مثال (٢) :- إذا كانت لدينا المصفوفة

	1	2	3
A	13	3	18

ابحث عن الرقم (٣٠)

N=3 ⟶ N+1=4

	1	2	3	4
A	13	3	18	30

Loc=1

A[Loc]#item ⟶ A[1]#30 ⟶ 13#30 ⟶ T.

Loc=2

A[Loc]#item ⟶ A[2]#30 ⟶ 3#30 ⟶ T.

Loc=3

A[Loc]#item ⟶ A[3]#30 ⟶ 30#30 ⟶ F.

إذا كان Loc=N+1 أظهر رساله بعدم وجود العنصر وإلا أكتب موقع العنصر وهو ال Loc ، وهنا تحقق هذا الشرط لذلك نطبع رساله تفيد بعدم وجود العنصر (٣٠).

Example:-

```
program linear_search1;
const m=10;
var a:array[1..m] of integer;
item,loc,i,n:integer;
begin
n:=m-1;
```

```
writeln('enter the data of array a:');
for i:=1 to n do
begin
writeln('enter a[',i,']: ');
readln(a[i]);
end;
writeln;
writeln('enter your item: ');
readln(item);
a[n+1]:=item;
loc:=1;
while a[loc]<>item do
loc:=loc+1;
if loc=n+1 then
writeln('the item is not found')
else
writeln('the item locationn is',loc);
readln;
end.
```

Output:-

```
enter the data of array a:
enter a[1]: 12
enter a[2]: 1
enter a[3]: 3
enter a[4]: 5
enter a[5]: 2
enter a[6]: 13
enter a[7]: 5
enter a[8]: 77
enter a[9]: 4
enter your item: 12
the item locationn is1
```

Example:-

اكتب برنامج لتخزين سجلات عدد من الطلاب داخل ملف فيزيائي ثم يقوم هذا البرنامج بطباعة سجلات الطلاب على شاشة التنفيذ ، ثم استخدم خوارزمية البحث الخطي للبحث عن سجل ما (حقل البحث هو الاسم) إذا وجده يطبع بإنانه وإذ لم يجده يطبع رسالة بعدم وجوده .

```
program linear_search2;
type
student=record
name:string[20];
no:integer;
faculty:string[20];
end;
var st:student;
c:char;
i:integer;
s:string[20];
f:file of student;

procedure linearsearch(s:string);
begin
reset(f);
read(f,st);
while (st.name<>s) and (not eof(f)) do
read(f,st);
with st do
if name=s then
begin
writeln('name....',name);
writeln('no....',no);
writeln('faculty....',faculty);
end
else
writeln('the record is not found');
close(f);
end;
begin{main program}
```

```
assign(f,'c:\file.doc');
rewrite(f);
writeln('enter the data or student:');
for i:=1 to 2 do
with st do
begin
writeln('enter the name,no,faculty for student',i);
readln(name);
readln(no);
readln(faculty);
write(f,st);
end;
close(f);
reset(f);
writeln('name   no      faculty');
while not eof(f) do
begin
read(f,st);
with st do
writeln(name,'   ',no,'   ',faculty);
end;
close(f);
repeat
writeln('enter the name of student to seatch:');
readln(s);
linearsearch(s);
writeln('enter y to continue or any key to stop:');
readln(c);
until c<>'y';
readln;
end.
```

Output:-

```
enter the data or student:
```

```
enter the name,no,faculty for student1

safiya

1

computer

enter the name,no,faculty for student2

fatima

2

shari3a

name     no      faculty

safiya   1       computer

fatima   2       shari3a

enter the name of student to search:

safiya

name....safiya

no....1

faculty....computer

enter y to continue or any key to stop:

q
```

- خوارزمية البحث الخطي هي من أبطئ الخوارزميات المستخدمة في البحث نسبة لأن هذه الخوارزمية تبحث من بداية الهيكل إلى نهاية وبالتالي تأخذ وقت طويل .

- عندما تقوم بتخزينه البيانات على أي هيكل من هياكل البيانات فإن هذه البيانات تخزن بصورة عشوائية .

- لا يشترط عملية الترتيب البيانات عند البحث عنها بواسطة " Linear Search " ، وتتم مقارنة العنصر المراد البحث عنه مع العنصر الأول المخزن فإذا تمت عملية المطابقة نقوم بالعمل المراد إجراءه على هذا العنصر ، أما إذا لم يتحصل عليه فسينتقل إلى العنصر التالي وهكذا إلى آخر عنصر .

تمرين(5):-

١- اكتب برنامج يقوم بتخزينه مجموعة من الأسماء على مصفوفة ثم يقوم البرنامج بالبحث عن اسم معين باستخدام البحث الخطي .

٢- اكتب برنامج يقوم بتخزينه سجلات ١٥ طلاب على مصفوفة ، ثم يقوم البرنامج بالبحث عن سجل طالب معين باستخدام حقل الرقم إذا وجده يطبع بياناته وإذا لم يجده يطبع رسالة بعد وجود هذا السجل مستخدماً البحث الخطي .

حل التمرين :-

(1)
```
program linear_search3
const m=10;
var a:array[1..m] of string;
loc,i,n:integer;
name:string;
begin
n:=m-1;
writeln('enter the data of array a:');
for i:=1 to n do
begin
write('enter a[',i,']:');
readln(a[i]);
end;
writeln('enter your name');
readln(name);
a[n+1]:=name;
loc:=1;
while a[loc]<>name do
loc:=loc+1;
if loc=n+1 then
writeln('the name is not found')
else
writeln('the name location is',loc);
readln;
end.
```

Output:-

```
enter the data of array a:

enter a[1]:safiya

enter a[2]:haneen

enter a[3]:fatima'

enter a[4]:alla

enter a[5]:banan

enter a[6]:yosof

enter a[7]:khlood

enter a[8]:mohamed

enter a[9]:ahmad

enter your name

haneen

the name location is2
```

(2) program linear_search4;

```
const m=3;
type
student=record
name:string[10];
no:integer;
faculty:string[10];
end;
var
a:array [1..m] of student ;
loc,item,i,n:integer;
begin
n:=m-1;
writeln('enter the data of  array a:');
```

```
for i:=1 to n do
with a[i] do
begin
write('enter name:');
readln(name);
write('enter no:');
readln(no);
write('enter faculty:');
readln(faculty);
end;
writeln('enter the number of student to search:');
readln(item);
a[n+1].no:=item;
loc:=1;
while a[loc].no<>item do
loc:=loc+1;
if loc=n+1 then
writeln ('the record is not found')
else
with a[loc] do
begin
writeln('name:',name);
writeln('no:',no);
writeln('faculty:',faculty);
writeln('the record location is',loc);
writeln('end..');
end;
readln;
end.
```

Output:-

```
enter the data of  array a:
enter name:ahmad
```

```
enter no:12
enter faculty:computer
enter name:mohamed
enter no:1
enter faculty:computer
enter the number of student to search:
١
name:mohamed
no:1
faculty:computer
the record location is2
end..
```

(٨-2) خوارزمية البحث الثنائي Binary Search :-

لإستخدام هذه الطريقة يجب أن تكون البيانات مرتبة ترتيباً تصاعدياً أو تنازلياً نلخص هذه الطريقة في أننا نقوم في كل مرحلة من مراحل البحث بمقارنة الهدف المراد البحث عنه مع العنصر الأوسط ، إذا كان متساويين يكون قد تحدد موقع العنصر المراد عنه ، أما إذا كان العنصر الأوسط أكبر من الهدف فيجب علينا البحث في النمو الأسفل من المصفوفة ،أما إذا كان العنصر الأوسط أصغر من الهدف فيجب علينا البحث في النصف الأعلى من المصفوفة ، نكرر هذه العملية بالنسبة للنصف الأسفل والنصف الأعلى ونتعامل معه كما لو كان مصفوفة قائمة بذاتها حتى نجد الهدف أو ينتهي البحث في دالة عدم عثورنا على الهدف .

الخوارزمية :-

١- احسب قيمة المؤشر الدليلي للعنصر الأوسط .
٢- إذا كان العنصر الأوسط مساوياً للهدف أرجع قيمة المؤشر الدليلي للعنصر الأوسط .
٣- إذا كان العنصر الأوسط أكبر من الهدف ابعث في المصفوفة الجزئية ذات المؤشرات الدليلية من موقع العنصر الأول إلى "موقع العنصر الأوسط-١" .
٤- إذا كان العنصر الأوسط أصغر من الهدف فابعث في المصفوفة الجزئية تارة المؤشرات الدليلة من " موقع العنصر الأوسط +١ " الى موقع العنصر الأخير .

مثال توضيحي:-

	1	2	3	4	5	6	7	8
A	1	3	9	21	33	40	90	99

- موقع العنصر الأوسط= <u>موقع العنصر الأول+الموقع الأخير</u>
  2
- المطلوب البحث عن العنصر (٣٣)
  موقع العنصر الوسط $(1+8)/2=4$
  Target=33

If A[4]=target ⟶ 21=33 ⟶ false.

- ابحث عن الرقم (٣٣) في المصفوفة

	5	6	7	8
A	33	40	90	99

موقع العنصر الأوسط $(5+8)/2=6$

If A[6]=target ⟶ 40=30 ⟶ false.

- ابحث عن الرقم (٣٣) في المصفوفة

5 6

A	33	40

موقع العنصر الوسط (5+6)/2=5

If A[5]=target ⟶ 33=33 ⟶ True.

إذاً العنصر (٣٣) موجود في الموقع ٥.

Example:-

```
program binary_search1;
type
data=array [1..10] of integer;
var i,b,target:integer;
a:data;

function binarysearch(var a:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1;
last:=10;
repeat
midd:=(first+last) div 2;
if a[midd]>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd]=target);
if a[midd]=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin{main program}
writeln('enter the data of array a:');
```

```
for i:=1 to 10 do
readln(a[i]);
writeln('enter the target:');
readln(target);
b:=binarysearch(a,target);
writeln('location of target=',b);
writeln('target=',a[b]);
readln;
end.
```

Output:-

```
enter the data of array a:
١٢
٣٣
١٢
٣٣
٦٧
٨٦
٤٤
٢٢
٧٨
٩٦
enter the target:
١٢
location of target=1
target=12
```

- خوارزميات البحث السريع لا أستطيع استخدامها داخل هيكل معين إلا إذا كانت البيانات مرتبة ، وإذا أردت أن أبحث والبيانات غير مرتبة لا أستطيع الا استخدام خوارزميات البحث الخطي وهذه من أنواع البحث الخاصة بالمصفوفات المرتبة وهو البحث الثنائي .
- عندما نتحدث عن خوارزمية البحث الثنائي يتضح لنا أنها أسرع .
- هذه الخوارزمية لا أستطيع استخدامها ما لم أرتب العناصر المخزنة تصاعدياً أو تنازلياً وهذه الخوارزمية يفترض أن تسبقها خوارزمية الترتيب ، أو لابد أن يقوم المستخدم بتخزين هذه البيانات بصورة مرتبة سواء أكان تصاعدياً أو تنازلياً .
- عندما نتحدث عن البحث في المصفوفة فإننا نتحدث عن المصفوفة المرتبة تصاعدياً أو تنازلياً أكبر من أو تساوي القيمة قيد البحث " الهدف " ، وذلك عندما تكون المصفوفة مرتبة تصاعدياً ، وليس هناك داع لفحص يأتي المصفوفة ، لأن الباقي من العنصر سيكون بالتأكيد أكبر من القيمة المطلوبة .
- وطريقة البحث هنا هي أننا لدينا مصفوفة تم تخزين عليها أعداد صحيحة أو أي نوع آخر من أنواع البيانات وهذه البيانات مرتبة ترتيباً تصاعدياً ، أولاً تقوم بتحديد العنصر الموجود في وسط المصفوفة وفي كل مرحلة من مراحل البحث تقوم بمقارنة الهدف " target " مع العنصر الأوسط " middle element " ويكون لدينا ثلاث احتمالات :-

١. إما أن يكون الـtargat مساوياً العنصر الموجود في الوسط وتكون بذلك قد انتهينا من العمل وأننا قمنا بتحيد الـLocation بالنسبة الـtarget .
٢. وإما أن يكون العنصر الأوسط أكبر من الـ target فهذا يعني أننا سنقوم بتقسيم المصفوفة الى جزئية وفي هذه الحالة سيتم البحث من بداية المصفوفة الى العنصر الموجود قبل العنصر الأوسط المصفوفة الجزئية وبهذا يتم عملية البحث في عدد من العناصر وبالتالي يكون البحث أسرع ، أي يتم البحث في الجزء الأسفل من المصفوفة .
٣. وإما أن يكون العنصر الأوسط أصغر من الـ target فإننا سنقوم بالبحث من العنصر التالي للعنصر الأوسط وستقوم بتكرار هذه العملية لكل مصفوفة جزئية الى أن نجد الهدف أو نطبع رسالة تقيد بعدم وجوده ، أي أن البحث سيكون في الجزء العلوي من المصفوفة .

تمرين (6):-

١- اكتب برنامج يعمل على تخزين مجموعة من الأسماء على مصفوفة مع مراعاة ترتيب الأسماء بمس حروف الهجاء ، والبحث عن اسم معين وإظهار موقعة باستخدام البحث الثنائي وإذا لم يجده يطبع رسالة تفيد ذلك

٢- اكتب برنامج يخزن سجلات ٥ موظفين على مصفوفة حيث بيانات الموظف هي الأسم ، الرقم ، المرتبه ، العضو وهنا يفترض مراعاة الترتيب في نقل الرقم سواء أدان ترتيب الارقام ترتيب تصاعدي أو ترتيب تنازلي وطباعة كافة السجلات على شاشة التنفيذ ، ثم يقوم البرنامج بالبحث عن سجل موظف معين باستخدام الرقم ويكون البحث :

أ/ البحث عن السجل الأول .

ب/ البحث عن سجل الموظف الأخير .

ج/ البحث عن سجل الموظف الموجود في وسط المصفوفة .

وبذلك يكون لدينا أكثر من شاشة output واحدة وذلك حتى يتسنى لنا التأكد من صحة البرنامج بصورة صحيحة .

حل التمرين :-

(1)
```
program binary_search2;
type
data=array [1..10] of string;
var i,b:integer;
target:string;
a:data;

function binarysearch(var a:data;target:string):integer;
var first,last,midd:integer;
begin
first:=1;
last:=10;
repeat
midd:=(first+last) div 2;
if a[midd]>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd]=target);
if a[midd]=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin{main program}
writeln('enter the data of array a:');
for i:=1 to 10 do
readln(a[i]);
writeln('enter the target:');
readln(target);
b:=binarysearch(a,target);
```

```
writeln('location of target=',b);
writeln('target=',a[b]);
readln;
end.
```

Output:-

```
enter the data of array a:
safiya
najeh
banan
haneen
khlood
fatima
yosof
riad
rashed
ibraheem
enter the target:
banan
location of target=3
target=banan
```

(2) program binary_search3;

```
type
employee=record
name:string[20];
no:integer;
salary:real;
age:integer;
end;
data=array [1..5] of employee;
```

```
var i,b,target:integer;
a:data; emp:employee;

function binarysearch(var a:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1;
last:=5;
repeat
midd:=(first+last) div 2;
if a[midd].no>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd].no=target);
if a[midd].no=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin{main program}
writeln('enter the data of employees:');
for i:=1 to 5 do
begin
write('enter name:');
readln(a[i].name);
write('enter no:');
readln(a[i].no);
write('enter salary:');
readln(a[i].salary);
write('enter age:');
readln(a[i].age);
end;
writeln('the data of employees are:');
writeln('name     no       salary       age');
```

```
with emp do
for i:=1 to 5 do
writeln(a[i].name,'     ',a[i].no,'     ',a[i].salary,'       ',a[i].age);
writeln;
writeln('enter the no of employee to search:');
readln(target);
b:=binarysearch(a,target);
writeln('location of target=',b);
writeln('the data of this employee is:');
with emp do
begin
writeln('name        no        salary        age');
writeln(a[b].name,'       ',a[b].no,'      ',a[b].salary,'
',a[b].age);
end;
readln;
end.
```

Output:-

```
enter the data of employees:
enter name:mohamed
enter no:2
enter salary:2323.1232
enter age:12
enter name:ahmad
enter no:1
enter salary:3434.543
enter age:54
enter name:yosof
enter no:3
enter salary:345.443
enter age:33
```

```
enter name:khaled
enter no:4
enter salary:2324.442
enter age:30
enter name:rashed
enter no:5
enter salary:4455.3433
enter age:40
the data of employees are:
name        no       salary                 age
mohamed     2        2.3231232000E+03       12
ahmad       1        3.4345430000E+03       54
yosof       3        3.4544300000E+02       33
khaled      4        2.3244420000E+03       30
rashed      5        4.4553433000E+03       40

enter the no of employee to search:
٤
location of target=4
the data of this employee is:
name          no         salary               age
khaled        4          2.3244420000E+03     30
```

****

الوحدة التاسعه

خوارزميات الترتيب

Sorting Algorithms

(٩-١) خوارزميات الترتيب الداخلي Internal Sorting Algorithms :-

(٩-١-١) الترتيب بالفقاع Bubble Sort :-

- في هذا النوع تطفو أصغر عناصر المصفوفة في أعلى المصفوفة .
- نفترض أن لدينا مصفوفة A وعدد عناصرها N في هذا النوع من الترتيب نتبع مايلي :

١- قارن بين العنصر الأول مع العنصر الثاني بحيث نضعهم في الترتيب A[1]<A[2] ثم نقارن العنصر الثاني مع العنصر الثالث ونضعهم في الترتيب A[2]<A[3] وتستمر هذه العملية حتى آخر عنصر من A[n-1]<A[n] وعند الانتهاء من هذه الخطوة يكون أكبر عنصر قد وضع في آخر المصفوفة A[n] .

٢- كرر الخطوة (١) بحيث تنتهي المقارنة عند العنصر قبل الأخير وعندها يكون هو أكبر عنصر في المصفوفة بعد العنصر A[n] .

٣- كرر الخطوة (2) n -1 مرة حتى تصل لمقارنة العنصر الأول مع العنصر الثاني A[1]<A[2] وعندما تكون المصفوفة قد رتبت.

خوارزمية الترتيب بالفقاع :-

١- كرر الخطوتين 3,2 بحيث k=1 to n-1 .

٢- أجعل j = 1 .

٣- كرر الخطوتين أ و ب بحيث j <= (n – k) :-

أ / إذا كان A[j]>A[j+1] بدل العنصرين مع بعض.

ب/ أجل j = j +1 .

٤- انهي التكرار في الخطوة (٣).

٥- انهي التكرار في الخطوة (١).

- في الترتيب بالفقاع ممكن أن نرتب تصاعدياً أو تنازلياً ولكن عادة ما نجل العناصر الصغيرة في بداية المصفوفة أي ترتيبها ترتيب تصاعدي .

الشكل التالي يوضح لنا هذه الخطوات :

1	2	3	4	5	6
7	12	8	17	4	5
7	12	8	17	4	5
7	8	12	17	5	5
7	8	12	17	4	5
7	8	12	4	17	5
7	8	12	4	5	17
7	8	12	4	5	17
7	8	12	4	5	17
7	8	4	12	5	17
7	8	4	5	12	17
7	8	4	5	12	17
7	4	8	5	12	17
7	4	5	8	12	17
4	7	5	8	12	17
4	5	7	8	12	17
4	5	7	8	12	17

- الملفات النصية هي الملفات التى تقرأ كاملة وتجرى عليها عملية المعالجة كاملة .
- حتى نستفيد من الترتيب لابد أن نرتب البيانات لأن الأصل في تخزين البيانات تخزين عشوائي .
- هذه الخورزمية بسيطة تقوم بترتيب البيانات الموجودة داخل ذاكرة الحاسوب والذاكرة دائماً حجمها صغير وبالتالي يتم إستخدام هذه الخوارزمية للترتيب الداخلي ، هنالك عدة من خوارزميات الترتيب الداخلية وكثرة هذه الخوارزميات هي للحصول على أفضل خوارزمية للترتيب وكذلك أكثر كفاءة أي تستخدم وقت قصر جداً لترتيب البيانات ولكن خوارزميات

الترتيب بالفقاع هي من أسوأ خوارزميات الترتيب لأنها تأخذ زمن طويل جداً حيث أنها تقوم بمقارنة العنصر الأول مع العنصر الثاني ونضع A[2]<A[1] وإذا كان بالعكس نقوم بترتيبها وهكذا الى آخر عنصر .

Example:-

```
program bubble1;
const n=10;
var a:array[1..n] of integer;
i,j,k,temp:integer;
begin
writeln('enter the data of array a:');
for i:=1 to n do
readln(a[i]);
writeln('the data befor sort is:');
for i:=1 to n do
write(a[i],' ');
writeln;
for k:=1 to n-1 do
begin
j:=1;
while j<=(n-k) do
begin
if a[j]>a[j+1] then
begin
temp:=a[j];
a[j]:=a[j+1];
a[j+1]:=temp;
end;
j:=j+1;
end;
end;
writeln('the data after sort is:');
for i:=1 to n do
write(a[i],' ');
readln;
end.
```

Output:-

```
enter the data of array a:
٤
٢٣
٦٧
٢
٥
٩
١
٧٧
٨
١١
the data befor sort is:
١١ ٨ ٧٧ ١ ٩ ٥ ٢ ٦٧ ٢٣ ٤
the data after sort is:
٧٧ ٦٧ ٢٣ ١١ ٩ ٨ ٥ ٤ ٢ ١
```

Example:-

برنامج يستخدم الترتيب بالفقاع (Bubble Sort) لترتيب عدد من الأسماء في مصفوفة ومن ثم تخزينها في ملف فيزيائي.

```
program bubble2;
var n:array[1..5] of string;
i,j,k:integer;
temp:string; f:text;
begin
assign(f,'c:\aaa.txt');
rewrite(f);
writeln('enter the names:');
for i:=1 to 5 do
readln(n[i]);
for k:=1 to 5 do
for j:=1 to 5-k do
```

```
if n[j]>n[j+1] then
begin
temp:=n[j];
n[j]:=n[j+1];
n[j+1]:=temp;
end;
writeln('the sort is:');
for i:=1 to 5 do
begin
writeln(n[i]);
write(f,n[i]);
end;
close(f);
readln;
end.
```

Output:-

```
enter the names:
fatima
safiya
banan
alaa
haneen
the sort is:
alaa
banan
fatima
haneen
safiya
```

تمرين (٧):-

- اكتب برنامج يقوم بتخزين سجلات 3 موظفين على مصفوفة حيث بيانات الموظف هي (الاسم nam ، الرقم id – no ، المرتبsalary ، المكان place ، الدرجة الوظيفية scale) ،

ثم يقوم البرنامج بطباعة سجلات جميع الموظفين على شاشة التنفيذ ، ثم يتضمين 3 برامج فرعية داخل هذا البرنامج :

١- برنامج فرعي يبحث مستخدماً الـ linear search حيث حقل البحث هو الرقم ثم أقوم بتغير هذه الخوارزمية بحيث الا أضف العنصر المراد البحث عنه في آخر المصفوفة وأول ما يجد العنصر المطلوب يقوم بإنهاء عملية البحث مباشرة .

٢- برنامج فرعى يقوم بترتيب السجلات بإستخدام حقل الرقم مستخدماً ال Bubble sort ثم يقوم بطابعة سجلات الموظفين بعد الترتيب .

٣- برنامج فرعي يبحث مستخدماً الـ binary search حيث حقل البحث هو الرقم ، إذا وجده يقوم بطباعة بياناته ثم يقوم باسناد القيمة صفر الى كل حقوله ثم يقوم بطباعة بيانته مرة أخرى .

ثم يقوم البرنامج بطباعة سجلات جميع الموظفين بعد إستخدام الـBinary search .

حل التمرين:-

```
program bubble3;
const m=4;
type
employee=record
name:string[20];
no:integer;
salary:real;
place:string[20];
scale:string[20];
end;
data=array[1..m]of employee;
var a:data; temp,emp:employee;
loc,item,k,j,i,s,b,target,l:integer;

procedure linearsearch(item:integer);
begin
l:=m-1;
a[l+1].no:=item;
loc:=1;
while a[loc].no<>item do
loc:=loc+1;
if loc=l+1 then
writeln('this record is not found')
```

```
else
begin
writeln('the record location is by linear search is:',l);
writeln('the data of this employee is:');
writeln('name       no       salary        place         scale');
with emp do
writeln(a[loc].name,'       ',a[loc].no,'       ',a[loc].salary,'
',a[loc].place,'       ',a[loc].scale);
end;
end;

function bubblesort(var a:data):integer;
begin
for k:=1 to 3 do
for j:=1 to 3-k do
if a[j].no>a[j+1].no then
begin
temp:=a[j];
a[j]:=a[j+1];
a[j+1]:=temp;
end;
writeln;
writeln('the sort of record is:');
writeln('name       no       salary        place         scale');
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
end;

function binarysearch(var a:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1; last:=3;
repeat
midd:=(first+last) div 2;
if a[midd].no>target then
```

```
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd].no=target);
if a[midd].no=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin {main program}
writeln('enter the data of employees:');
for i:=1 to 3 do
begin
write('enter the name of employee',i,': ');
readln(a[i].name);
write('enter the no of employee',i,': ');
readln(a[i].no);
write('enter the salary of employee',i,': ');
readln(a[i].salary);
write('enter the place of employee',i,': ');
readln(a[i].place);
write('enter the scale of employee',i,': ');
readln(a[i].scale);
end;
writeln('the data of employee are:');
writeln('name       no       salary        place         scale');
with emp do
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
writeln;
writeln('enter the no of employee to search by linear search:');
readln(item);
linearsearch(item);
writeln;
```

```
s:=bubblesort(a);
writeln;
writeln('enter the no of employee to search by binary search:');
readln(target);
b:=binarysearch(a,target);
if a[b].no=target then
begin
writeln('the record location by binary search is=',b);
writeln('the data of this employee is:');
writeln('name       no       salary        place         scale');
with emp do
begin
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'       ',a[b].scale);
a[b].name:='0';
a[b].no:=0;
a[b].salary:=0;
a[b].place:='0';
a[b].scale:='0';
writeln;
writeln('name       no       salary        place         scale');
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'      ',a[b].scale);
end;
end
else
writeln('this record is not found');
writeln;
writeln('the data of employee are:');
writeln('name       no       salary        place         scale');
with emp do
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
readln;
end.
```

Output:-

```
enter the data of employees:
enter the name of employee1: safiya
enter the no of employee1: 4
enter the salary of employee1: 2323.223
enter the place of employee1: iraq
enter the scale of employee1: a
enter the name of employee2: najeh
enter the no of employee2: 1
enter the salary of employee2: 234.232
enter the place of employee2: iraq
enter the scale of employee2: b
enter the name of employee3: banan
enter the no of employee3: 3
enter the salary of employee3: 34534.343
enter the place of employee3: tunis
enter the scale of employee3: c
the data of employee are:
name        no       salary                 place         scale
safiya      4        2.3232230000E+03       iraq          a
najeh       1        2.3423200000E+02       iraq          b
banan       3        3.4534343000E+04       tunis         c

enter the no of employee to search by linear search:
١
the record location is by linear search is:2
the data of this employee is:
name        no       salary                 place         scale
najeh       1        2.3423200000E+02       iraq          b


the sort of record is:
name        no       salary                 place         scale
najeh       1        2.3423200000E+02       iraq          b
banan       3        3.4534343000E+04       tunis         c
safiya      4        2.3232230000E+03       iraq          a
```

```
enter the no of employee to search by binary search:
٣
the record location by binary search is=2
the data of this employee is:
name        no       salary                place         scale
banan       3        3.4534343000E+04      tunis         c

name                no       salary      place     scale
٠.٠٠٠٠٠٠٠٠٠٠        ٠        ٠E+00       0         0

the data of employee are:
name               no     salary              place         scale
najeh              1      2.3423200000E+02    iraq          b
٠ ٠.٠٠٠٠٠٠٠٠٠٠     ٠      E+00                0             0
safiya             4      2.3232230000E+03    iraq          a
```

(٢-١-٩) الترتيب بالإختيار Selection Sort :-

- لنفرض أن لدينا مجموعة من الأعداد عددها (n) ونريد ترتيبها تصاعدياً باستعمال الترتيب بالاختيار .
- الفكرة الأساسية لهذه الطريقة هو البحث عن أصغر عنصر في المصفوفة واستبداله مع العنصر الأول a[1] ، ثم نبحث عن أصغر عنصر بين العناصر a[2] إلى a[n] وتقوم بتبديله مع العنصر الثاني a[2] ، ثم تأخذ العناصر من a[3] إلى a[n] ونبحث عن أصغرها وتبديله مع العنصر الثالث a[3] وهكذا حتى تنتهي من كل عناصر المصفوفة فنحصل على مصفوفة مرتبة .

الشكل التالي يوضح هذه الخطوات:-

	1	2	3	4	5	6
(1)	7	12	8	17	4	5
(2)	4	12	8	17	7	5
(3)	4	5	8	17	7	12
(4)	4	5	7	17	8	12
(5)	4	5	7	8	17	12
(6)	4	5	7	8	12	17

- تحديد العنصر الأصغر يعني تحديد ال Location له.

خوارزمية الترتيب بالاختيار :-

١- ضع i = 1 .
٢- ابحث عن أصغر عنصر من عناصر المصفوفة من a[i] الى a[n] .
٣- قم بتبديل العنصر الأصغر مع a[i] .
٤- أجعل i = i +1 .
٥- إذا كان i=n حيث (n) عدد عناصر المصفوفة فإن المصفوفة تكون قد رتبت والا أذهب الى الخطوة رقم (2).

- إذا أردت ألا أضيف العنصر المراد البحث عنه في ال linear search الى آخر المصفوفة وذلك لإستغلال المساحة نضيف Condition الى شرط حلقة while فيصبح الشرط كالتالي:-

```
While (a[loc]<>item) and (i<=n) do
```

Example:-

```
program select1;
const n=10;
var a:arrays[1..10] of integer;
i,j,k,s:integer;
begin
writeln('enter the data of array a:');
```

```
for i:=1 to 10 do
readln(a[i]);
writeln('the data befor sort is:');
for i:=1 to n do
write(a[i],' ');
for j:=1 to 10 do
begin
k:=j; s:=a[j];
for i:=j+1 to 10 do
begin
if a[i]<s then
begin
s:=a[i]; k:=i;
end;
end; {end of for i}
s:=a[j]; a[j]:=a[k]; a[k]:=s;
end; {for i}
writeln;
writeln('the data after sort is:');
for i:=1 to 10 do
write(a[i],' ');
readln;
end.
```

Output:-

```
enter the data of array a:
١٢
٤٥
٣
٢٣
١
٦٧
٥
```

```
٣٠
١١
٥١
the data befor sort is:
٥١ ١١ ٣٠ ٥ ٦٧ ١ ٢٣ ٣ ٤٥ ١٢
the data after sort is:
٦٧ ٥١ ٤٥ ٣٠ ٢٣ ١٢ ١١ ٥ ٣ ١
```

Example:-

```
program select2;
const n=10;
type
data=array[1..n] of integer;
var a:data;
i,j:integer;

procedure selectsort(var table:data;n:integer);
var maxposition:integer;
function find_maxpos(var table:data;n:integer):integer;
var i,maxpos:integer;
begin
maxpos:=1;
for i:=2 to n do
if t able[i]>table[maxpos] then
maxpos:=i;
find_maxpos:=maxpos;
end;
procedure exchange(var x,y:integer);
var temp:integer;
begin
temp:=x;
x:=y;
y:=temp;
end;
```

```
begin {procedure selectsort}
if n>1 then
begin
maxposition:=find_maxpos(table,n);
exchange(table[maxposition],table[n]);
selectsort(table,n-1);
end;
end; {procedure selectsort}

begin{main program}
writeln('enter the data of array a:');
for i:=1 to n do
readln(a[i]);
writeln('the data befor sort is:');
for i:=1 to n do
write(a[i],' ');
writeln;
selectsort(a,n);
writeln('the data after sort is:');
for i:=1 to n do
write(a[i],' ');
readln;
end.
```

Output:-

```
enter the data of array a:
١٢
٤٥
٣
٢٣
١
٦٧
٥
```

```
٣٠
١١
٥١
the data befor sort is:
٥١ ١١ ٣٠ ٥ ٦٧ ١ ٢٣ ٣ ٤٥ ١٢
the data after sort is:
٦٧ ٥١ ٤٥ ٣٠ ٢٣ ١٢ ١١ ٥ ٣ ١
```

Example:-

```
program select3;
const m=4;
type
employee=record
name:string[20];
no:integer;
salary:real;
place:string[20];
scale:string[20];
end;
data=array[1..m]of employee;
var a:data; emp,s:employee;
loc,item,k,j,i,b,target,n:integer;

procedure linearsearch(item:integer);
begin
n:=m-1;
a[n+1].no:=item;
loc:=1;
while (a[loc].no<>item) and (i<=n) do
loc:=loc+1;
if loc=n+1 then
writeln('this record is not found')
else
begin
```

```
writeln('the record location is by linear search is:',loc);
writeln('the data of this employee is:');
writeln('name       no       salary       place        scale');
with emp do
writeln(a[loc].name,'       ',a[loc].no,'       ',a[loc].salary,'
',a[loc].place,'       ',a[loc].scale);
end;
end;

procedure selectsort(var a:data);
begin
for j:=1 to 3 do
begin
k:=j; s:=a[j];
for i:=j+1 to 3 do
begin
if a[i].no<s.no then
begin
s:=a[i]; k:=i;
end;
end; {end of for i}
s:=a[j]; a[j]:=a[k]; a[k]:=s;
end; {for i}
writeln;
writeln('the sort of record is:');
writeln('name       no       salary       place        scale');
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
end;

function binarysearch(var a:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1; last:=3;
repeat
```

```
midd:=(first+last) div 2;
if a[midd].no>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd].no=target);
if a[midd].no=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin {main program}
writeln('enter the data of employees:');
for i:=1 to 3 do
begin
write('enter the name of employee',i,': ');
readln(a[i].name);
write('enter the no of employee',i,': ');
readln(a[i].no);
write('enter the salary of employee',i,': ');
readln(a[i].salary);
write('enter the place of employee',i,': ');
readln(a[i].place);
write('enter the scale of employee',i,': ');
readln(a[i].scale);
end;
writeln('the data of employee are:');
writeln('name       no       salary       place        scale');
with emp do
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
writeln;
writeln('enter the no of employee to search by linear search:');
readln(item);
```

```
linearsearch(item);
writeln;
selectsort(a);
writeln;
writeln('enter the no of employee to search by binary search:');
readln(target);
b:=binarysearch(a,target);
if a[b].no=target then
begin
writeln('the record location by binary search is=',b);
writeln('the data of this employee is:');
writeln('name       no       salary        place         scale');
with emp do
begin
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'       ',a[b].scale);
a[b].name:='0';
a[b].no:=0;
a[b].salary:=0;
a[b].place:='0';
a[b].scale:='0';
writeln;
writeln('name       no       salary        place         scale');
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'      ',a[b].scale);
end;
end
else
writeln('this record is not found');
writeln;
writeln('the data of employee are:');
writeln('name       no       salary        place         scale');
with emp do
for i:=1 to 3 do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
```

```
readln;
end.
```

Output:-

```
enter the data of employees:
enter the name of employee1: gege
enter the no of employee1: 3
enter the salary of employee1: 232.1212
enter the place of employee1: lebnan
enter the scale of employee1: e
enter the name of employee2: seera
enter the no of employee2: 2
enter the salary of employee2: 232.32
enter the place of employee2: dubai
enter the scale of employee2: a
enter the name of employee3: tema
enter the no of employee3: 9
enter the salary of employee3: 3223.223
enter the place of employee3: landon
enter the scale of employee3: c
the data of employee are:
name        no       salary                 place        scale
gege        3        2.3212120000E+02       lebnan       e
seera       2        2.3232000000E+02       dubai        a
tema        9        3.2232230000E+03       landon       c

enter the no of employee to search by linear search:
٢
the record location is by linear search is:2
the data of this employee is:
name        no       salary                 place        scale
seera       2        2.3232000000E+02       dubai        a


the sort of record is:
name        no       salary                 place        scale
```

```
seera        2        2.3232000000E+02        dubai        a
gege         3        2.3212120000E+02        lebnan       e
tema         9        3.2232230000E+03        landon       c
enter the no of employee to search by binary search:
٠
this record is not found

the data of employee are:
name         no       salary                  place        scale
seera        2        2.3232000000E+02        dubai        a
gege         3        2.3212120000E+02        lebnan       e
tema         9        3.2232230000E+03        landon       c
```

(3-١-٩) الترتيب بالإدخال Insertion Sort :-

- هذه الطريقة تختصر جزء من عمل طريقة الترتيب بالاختيار ، فنقوم أولاً بالبحث عن أصغر في المصفوفة ونحتفظ به في متغير آخر ثم نقوم بإزاحة للعناصر من a[1] إلى العنصر الذي يقع قبل مكان العنصر الأصغر الذي وجدناه ، فيتقدم كل عنصر خطوة الأمام ونضع العنصر الأصغر في a[1] ، ثم نكرر هذه العملية السابقة أخذين العناصر من a[2] وحث آخر عنصر في المصفوفة .

الشكل التالي يوضح هذه الخطوات:-

	1	2	3	4	5	6
(1)	7	12	8	17	4	5
(2)	4	7	12	8	17	5
(3)	4	5	7	12	8	17
(4)	4	5	7	8	12	17

خوارزمية الترتيب بالإدخال :-

١- ضع i =1 .
٢- ابحث أصغر عنصر من عناصر المصفوفة من a[i] وحتى a[n] .
٣- ضع العنصر الأصغر في المتغير x .
٤- اعمل إزاحة لجميع العناصر من a[i] وحتى العنصر الذي يقع قبل مكان العنصر الأصغر بتقديم كل منهم خطوة للأمام "Shift" .
٥- ضع a[i]:=x .
٦- ضع i =i +1 .
٧- إذا كان (i = n) فإن المصفوفة قد رتبت وإلا أذهب إلى الخطوة رقم (2) .

***Insert Sort in integer array:-**

```
program insert1;
var a:array [1..10] of integer;
i,j,k,c,n,min:integer;
begin
writeln('enter the number of elements:');
readln(n);
writeln('enter the elements of array a:');
for i:=1 to n do
begin
write('A',i,'=');
readln(a[i]);
end;
writeln('the elements of array A befor sorting are:');
for i:=1 to n do
write(a[i],' ');
writeln;
for i:=1 to n do
begin
k:=0;
min:=a[i];
for j:=i to n do
if min>a[j] then
begin
```

```
min:=a[j];
k:=j;
end;
for c:=k downto i do
a[c]:=a[c-1];
a[i]:=min;
end; {end of for i}
writeln('the elements of array A after sort is:');
for i:=1 to n do
write(a[i],' ');
readln;
end.
```

Output:-

```
enter the number of elements:
١٠
enter the elements of array a:
A1=21
A2=44
A3=1
A4=4
A5=88
A6=5
A7=32
A8=10
A9=11
A10=23
the elements of array A befor sorting are:
٢٣ ١١ ١٠ ٣٢ ٥ ٨٨ ٤ ١ ٤٤ ٢١
the elements of array A after sort is:
٨٨ ٤٤ ٣٢ ٢٣ ٢١ ١١ ١٠ ٥ ٤ ١
```

*Insert Sort in record:-

```
program insert2;
type
employee=record
name:string;
age:integer;
address:string;
end;
var emp:array[1..10]of employee;
i,j,k,c,n:integer;
min:employee;

begin{main program}
writeln('enter the numbers of employees:');
readln(n);
for i:=1 to n do
begin
writeln('enter the data of employee[',i,']:');
write('enter the name:');
readln(emp[i].name);
write('enter the age:');
readln(emp[i].age);
write('enter the address:');
readln(emp[i].address);
end;
writeln;
writeln('the data befor sorting are:');
writeln('name     age     address');
for i:=1 to n do
writeln(emp[i].name,' ',emp[i].age,' ',emp[i].address);
writeln;
for i:=1 to n do
begin
k:=0;
min:=emp[i];
for j:=i to n do
if min.age>emp[j].age then
```

```
begin
min:=emp[j];
k:=j;
end;
for c:=k downto i do
emp[c]:=emp[c-1];
emp[i]:=min;
end; {end of for i}
writeln('the data after sorting are:');
writeln('name     age     address');
for i:=1 to n do
writeln(emp[i].name,' ',emp[i].age,' ',emp[i].address);
readln;
end.
```

Output:-

```
enter the numbers of employees:
٣
enter the data of employee[1]:
enter the name:doha
enter the age:19
enter the address:3man
enter the data of employee[2]:
enter the name:noor
enter the age:20
enter the address:Irbed
enter the data of employee[3]:
enter the name:yosof
enter the age:5
enter the address:tunis
the data befor sorting are:
name     age     address
```

```
doha     19      3man
noor     20      Irbed
yosof    5       tunis

the data after sorting are:
name     age     address
yosof    5       tunis
doha     19      3man
noor     20      Irbed
```

*Insertion Sort in file:-

```
program insert3;
type
employee=record
name:string;
age:integer;
address:string;
end;
var emp:array[1..10] of employee;
i,j,k,c,n:integer;
temp,min:employee;
f:file of employee;

begin {main program}
assign(f,'c:\emp.dat');
rewrite(f);
writeln('enter the numbers of employees:');
readln(n);
writeln('enter the data of employees:');
for i:=1 to n do
begin
writeln('enter the data of employee[',i,']:');
write('enter the name:');
```

```
readln(emp[i].name);
write('enter the age:');
readln(emp[i].age);
write('enter the address:');
readln(emp[i].address);
write(f,emp[i]);
end;
writeln;
writeln('the data befor sorting are:');
reset(f);
writeln('name     age     address');
i:=1;
while not eof(f) do
begin
read(f,emp[i]);
with emp[i] do
writeln(name,' ',age,' ',address);
i:=i+1;
end;
writeln;
for i:=1 to n do
begin
k:=0;
min:=emp[i];
for j:=i to n do
if min.age>emp[j].age then
begin
min:=emp[j];
k:=j;
end;
for c:=k downto i do
emp[c]:=emp[c-1];
emp[i]:=min;
end;
rewrite(f);
for i:=1 to n do
```

```
write(f,emp[i]);
writeln('the data after sorting are:');
reset(f);
writeln('name     age     address');
i:=1;
while not eof(f) do
begin
read(f,emp[i]);
with emp[i] do
writeln(name,' ',age,' ',address);
i:=i+1;
end;
close(f);
readln;
end.
```

Output:-

```
enter the numbers of employees:
٢
enter the data of employees:
enter the data of employee[1]:
enter the name:mohamed
enter the age:30
enter the address:sudan
enter the data of employee[2]:
enter the name:ahmad
enter the age:20
enter the address:sudan

the data befor sorting are:
name      age     address
mohamed   30      sudan
```

```
ahmad     20      sudan


the data after sorting are:

name      age     address

ahmad     20      sudan

mohamed   30      sudan
```

New way in file:-

```
program insert4;
type
employee=record
name:string;
age:integer;
address:string;
end;
var emp:array[1..10] of employee;
i,j,k,c,n:integer;
temp,min:employee;
f:file of employee;

begin {main program}
assign(f,'c:\emp.dat');
rewrite(f);
writeln('enter the numbers of employees:');
readln(n);
writeln('enter the data of employees:');
for i:=1 to n do
begin
writeln('enter the data of employee[',i,']:');
write('enter the name:');
readln(emp[i].name);
write('enter the age:');
readln(emp[i].age);
write('enter the address:');
```

```
readln(emp[i].address);
write(f,emp[i]);
end;
writeln;
writeln('the data befor sorting are:');
reset(f);
writeln('name     age     address');
while not eof(f) do
begin
read(f,temp);
with temp do
writeln(name,' ',age,' ',address);
end;
writeln;
for i:=1 to n do
begin
k:=0;
min:=emp[i];
for j:=i to n do
if min.age>emp[j].age then
begin
min:=emp[j];
k:=j;
end;
for c:=k downto i do
emp[c]:=emp[c-1];
emp[i]:=min;
end;
rewrite(f);
for i:=1 to n do
write(f,emp[i]);
writeln('the data after sorting are:');
reset(f);
writeln('name     age     address');
while not eof(f) do
begin
```

```
read(f,temp);
with temp do
writeln(name,' ',age,' ',address);
end;
close(f);
readln;
end.
```

Output:-

```
enter the numbers of employees:
٢
enter the data of employees:
enter the data of employee[1]:
enter the name:walaa
enter the age:22
enter the address:sudan
enter the data of employee[2]:
enter the name:olaa
enter the age:33
enter the address:sudan

the data befor sorting are:
name      age      address
walaa     22       sudan
olaa      33       sudan

the data after sorting are:
name      age      address
walaa     22       sudan
olaa      33       sudan
```

(٤-١-٩) الترتيب بالتجزئه Shell Sort :-

تنفذ هذه الطريقة في عدة مراحل ، فلو افترضنا عدد عناصر المصفوفه "6 " فإن هذه المراحل تنفذ كمايلي

- **المرحلة الأولي : -**

d=(b+1) /2=3

يقارن العنصر الأول مع الرابع (القفز 3) والثاني مع الخامس والثالث مع السادس وتنفذ عملية المبادلة إذا لزم الأمر .

- **المرحلة الثانية :-**

d=(3+1)/2=2

يقارن العنصر الأول مع الثالث (القفز 2) والثاني مع الرابع والثالث مع الخامس والخامس مع السادس وتنفذ عملية المبادلة إذا لزم الأمر .

- **المرحلة الثالثة :-**

d=(2+1)/2=1

يقارن العنصر الأول مع الثاني (القفز 1) والثالث مع الرابع وهكذا .

- **المرحلة الأخيرة :-**

متابعة الترتيب حتى يصبح (d=1) وحتى لا تحصل عملية مبادلة .

المثال التالي يوضح ذلك وهنا الترتيب تنازلي:-

Array at beginning:	84	69	76	86	94	91	d
After pass#1:	86	94	91	84	69	76	3
After pass#2:	91	94	86	84	69	76	2
After pass#3:	94	91	86	84	76	69	1
After pass#4(done):	94	91	86	84	76	69	1

Example:-

```
program shell1;
type
data=array[1..10] of integer;
var x:data;
```

```
i,n:integer;

procedure shell_sort(var a:data;n:integer);
var flag,d,temp:integer;
begin
flag:=1; d:=n;
while (flag<>0) or (d>1) do
begin
flag:=0; d:=(d+1) div 2;
for i:=1 to n-d do
begin
if a[i+d]>a[i] then
begin
temp:=a[i+d];
a[i+d]:=a[i];
a[i]:=temp;
flag:=1;
end;
end;
end;
end;

begin {main program}
writeln('enter the number of elements:');
readln(n);
writeln('enter the elements:');
for i:=1 to n do
readln(x[i]);
writeln('the elements befor sort is:');
for i:=1 to n do
write(x[i],' ');
writeln;
shell_sort(x,n);
writeln('the elements after sort is:');
for i:=1 to n do
write(x[i],' ');
```

```
readln;
end.
```

Output:-

```
enter the number of elements:
٥
enter the elements:
١٢
٤٤
٥
٦٧
١٠
the elements befor sort is:
١٠ ٦٧ ٥ ٤٤ ١٢
the elements after sort is:
٥ ١٠ ١٢ ٤٤ ٦٧
```

Example:-

```
program shell2;
type
employee=record
name:string;
no:integer;
address:string;
end;
data=array[1..10]of employee;
var a:data;
i,j,k,c,n:integer;
procedure shell_sort(var a:data;n:integer);
var flag,d:integer;
temp:employee;
begin
flag:=1; d:=n;
while (flag<>0) or (d>1) do
begin
flag:=0; d:=(d+1) div 2;
```

```
for i:=1 to n-d do
begin
if a[i+d].no>a[i].no then
begin
temp:=a[i+d];
a[i+d]:=a[i];
a[i]:=temp;
flag:=1;
end;
end;
end;
end;

begin{main program}
writeln('enter the numbers of employees:');
readln(n);
for i:=1 to n do
begin
writeln('enter the data of employee[',i,']:');
write('enter the name:');
readln(a[i].name);
write('enter the no:');
readln(a[i].no);
write('enter the address:');
readln(a[i].address);
end;
writeln;
writeln('the data befor sorting are:');
writeln('name     no     address');
for i:=1 to n do
writeln(a[i].name,'  ',a[i].no,'  ',a[i].address);
writeln;
shell_sort(a,n);
writeln('the data after sorting are:');
writeln('name     no     address');
for i:=1 to n do
```

```
writeln(a[i].name,' ',a[i].no,' ',a[i].address);
readln;
end.
```

Output:-

```
enter the numbers of employees:
٣
enter the data of employee[1]:
enter the name:ahmad
enter the no:8
enter the address:sudan
enter the data of employee[2]:
enter the name:samiya
enter the no:2
enter the address:sudan
enter the data of employee[3]:
enter the name:3adel
enter the no:5
enter the address:sudan

the data befor sorting are:
name     no      address
ahmad    8       sudan
samiya   2       sudan
٣adel    5       sudan

the data after sorting are:
name     no      address
ahmad    8       sudan
٣adel    5       sudan
samiya   2       sudan
```

Example:-

```
program shell3;
type
employee=record
name:string;
```

```
no:integer;
address:string;
end;
data=array[1..10] of employee;
var emp:data;
i,j,k,c,n:integer;
f:file of employee;

procedure shell_sort(var a:data;n:integer);
var flag,d:integer;
temp:employee;
begin
flag:=1; d:=n;
while (flag<>0) or (d>1) do
begin
flag:=0; d:=(d+1) div 2;
for i:=1 to n-d do
begin
if a[i+d].no>a[i].no then
begin
temp:=a[i+d];
a[i+d]:=a[i];
a[i]:=temp;
flag:=1;
end;
end;
end;
end;

begin {main program}
assign(f,'c:\emp.dat');
rewrite(f);
writeln('enter the numbers of employees:');
readln(n);
writeln('enter the data of employees:');
for i:=1 to n do
```

```
begin
writeln('enter the data of employee[',i,']:');
write('enter the name:');
readln(emp[i].name);
write('enter the no:');
readln(emp[i].no);
write('enter the address:');
readln(emp[i].address);
write(f,emp[i]);
end;
writeln;
writeln('the data befor sorting are:');
reset(f);
writeln('name     no     address');
i:=1;
while not eof(f) do
begin
read(f,emp[i]);
with emp[i] do
writeln(name,' ',no,' ',address);
i:=i+1;
end;
writeln;
shell_sort(emp,n);
rewrite(f);
for i:=1 to n do
write(f,emp[i]);
writeln('the data after sorting are:');
reset(f);
writeln('name     no     address');
i:=1;
while not eof(f) do
begin
read(f,emp[i]);
with emp[i] do
writeln(name,' ',no,' ',address);
```

```
i:=i+1;
end;
close(f);
readln;
end.
```

Output:-

```
enter the numbers of employees:
٢
enter the data of employees:
enter the data of employee[1]:
enter the name:sayaf
enter the no:2
enter the address:tunis
enter the data of employee[2]:
enter the name:omar
enter the no:23
enter the address:tunis

the data befor sorting are:
name     no     address
sayaf    2      tunis
omar     23     tunis

the data after sorting are:
name     no     address
omar     23     tunis
sayaf    2      tunis
```

(٥-١-٩) الترتيب السريع Quick Sort :-

- تُعدّ هذه الخوارزمية من أفضل وأسرع خوارزميات الترتيب ، إذ أنها تستخدم تقنية التصميم في عملها والتي تؤدي إلى تقسيم القائمة المطلوب ترتيبها إلى قسمين يفصل بينها عنصر منتخب (pivot) ، حيث تكون قيم القسم الأول أصغر أو تساوي قيمة العنصر

المنتخب وتوضع يساره، وقيم القسم الثاني أكبر من قيمة العنصر المنتخب وتوضع على يمينه. وفي القوائم الكبيرة يمكن تقسيم كل قسم إلى قسمين أخريين وهكذا إلى أن يتم ترتيب جميع الأقسام ومن ثم دمجهم مع بعض لتكوين قائمة مرتبة.

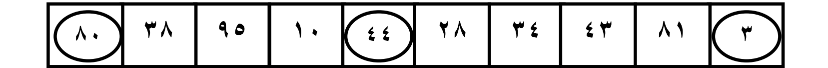
أي أن خوارزمية الترتيب السريع تتكون من ثلاث مراحل وهي :

١. اختيار العنصر المنتخب (Choose a pivot value).
٢. التقسيم (Partition).
٣. الترتيب (Sort).

- إن عملية اختيار العنصر المنتخب هي من أصعب أجزاء هذه الخوارزمية، لأنه في العادة يتم تقسيم القائمة إلى نصفين متساويين بغض النظر عن قيمة العنصر المنتخب. وهذا قد يتسبب في زيادة وقت التنفيذ إذ لم تكن القيمة المنتخبة في موقعها الصحيح ، أي ربما تكون في موقع معاكس لعملية الترتيب. والحل البديل لهذه الحالة هو اختيار ثلاث قيم من القائمة غير المرتبة وغالباً ما تكون القيمة الأولى والوسطى والأخيرة ومن ثم اختيار القيمة الأوسط بين تلك القيم.

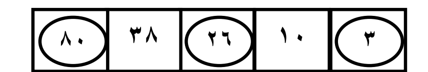
إذا كانت لدينا القوائم التالية:

تم اختيار القيمة (٤٤) للقائمة التالية:-



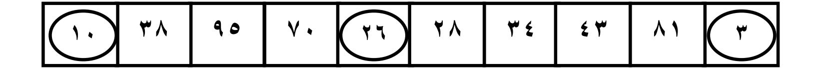
٨٠	٣٨	٩٥	١٠	٤٤	٢٨	٣٤	٤٣	٨١	٣

واختيار القيمة (٢٦) للقائمة التالية:-



٨٠	٣٨	٢٦	١٠	٣

واختيار القيمة (١٠) للقائمة التالية:-



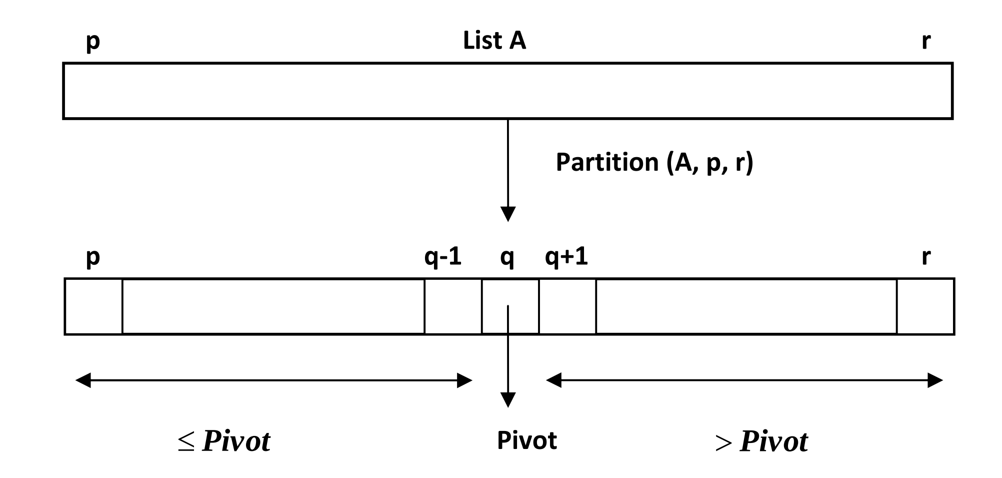
١٠	٣٨	٩٥	٧٠	٢٦	٢٨	٣٤	٤٣	٨١	٣

وبعد اختيار قيمة العنصر المنتخب يتم تقسيم القائمة غير المرتبة إلى قائمتين (Two sub-list).

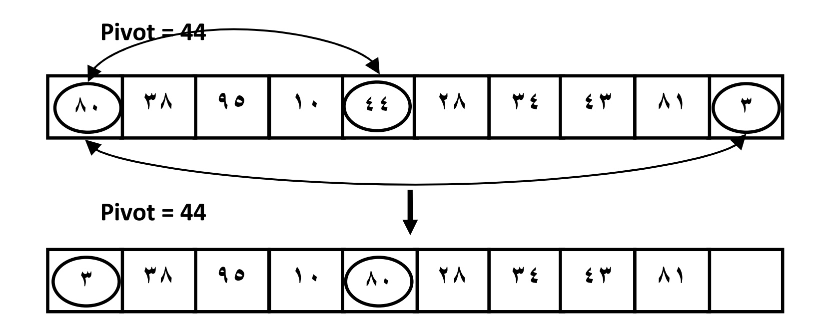
الأولى: تضم العناصر الواقعة بين العنصر الأول (p) والعنصر المنتخب (q).

والثانية : تضم العناصر الواقعة بعد العنصر المنتخب (q+1) والعنصر الأخير (r).

وكما موضح في الشكل الآتي :



وبعد اختيار العنصر المنتخب نقوم بوضع القيمة الصغرى مكان العنصر الأول والقيمة الكبرى مكان القيمة الوسطى. **وكما موضح في المثال الآتي :**

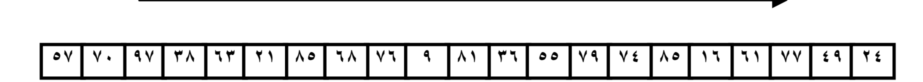


ثم نبدأ بعملية ترتيب القائمة والتي تعمل على نقل جميع العناصر الكبيرة إلى يمين القائمة (أي يمين العنصر المنتخب) وأيضاً نقل جميع العناصر الصغيرة إلى يسار القائمة. وهذه العملية تحتاج إلى مؤشرين (Two indices) الأول (*i*) وهو الأيسر ، يتحرك من اليسار إلى اليمين. والثاني (*j*) وهو الأيمن ، ويتحرك من اليمين إلى اليسار حسب **الخوارزمية الآتية:**

١. عندما يكون (*i*) يسار (*j*) ، يتحرك (*i*) نحو اليمين متخطياً كل العناصر التي تكون قيمها أصغر من قيمة العنصر المنتخب.
٢. إذا وجد أي عنصر أكبر من قيمة العنصر المنتخب ، يتوقف المؤشر(*i*).
٣. عندما يكون (*j*) يمين (*i*) ، يتحرك (*j*) نحو اليسار متخطياً كل العناصر التي تكون قيمها أكبر من قيمة العنصر المنتخب.
٤. إذا وجد أي عنصر أصغر من قيمة العنصر المنتخب ، يتوقف المؤشر(*j*).
٥. عندما يتوقف كل من (*i*) و (*j*) فهذا يعني أن هناك عملية تبديل.
٦. عندما يتقاطع (Cross) المؤشر (*i*) مع المؤشر (*j*) تتوقف عملية المقارنة وتستبدل قيمة العنصر المنتخب بقيمة العنصر الذي حدثت عنه عملية التقاطع.

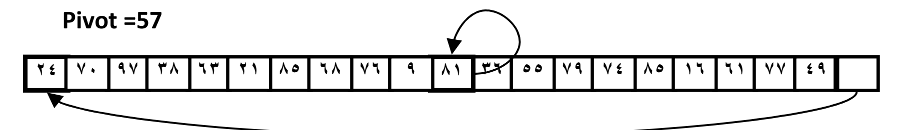
٧. بعد تحديد القيمة الجديدة للعنصر المنتخب ، تجري عملية الترتيب لكل جزء على حدا.
٨. بشكل تكراري ، يتم اختيار عنصر منتخب لكل جزء وتعاد الخطوات من ١-٧.

- **مثال تطبيقي:**
  - ❖ **رتب القائمة الآتية ترتيباً تصاعدياً؟**

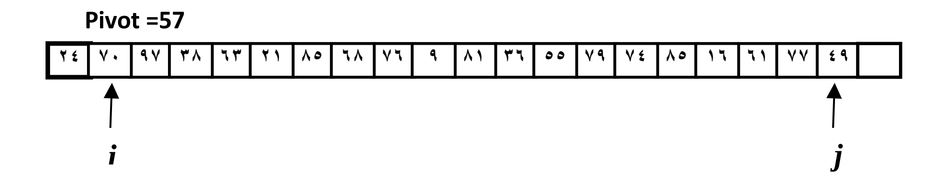


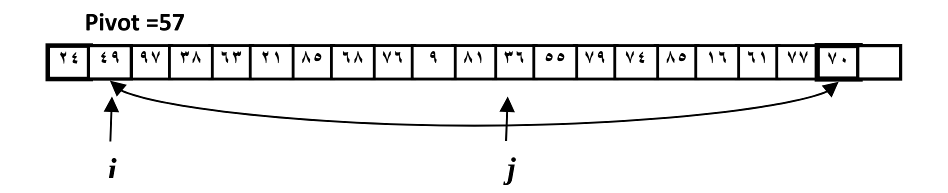
❖ **الحل :**

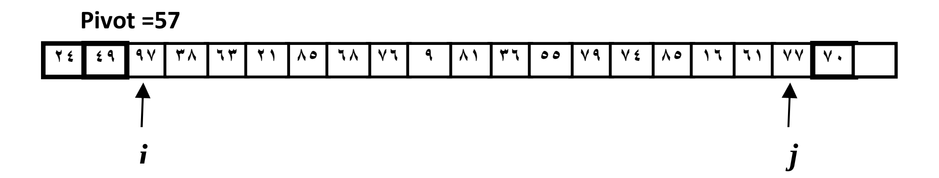
١. اختيار العنصر المنتخب باستخدام طريقة القيمة الوسطى بين ثلاث قيم $[57,81,24]$.
والعنصر سيتكون القيمة (57).
٢. ترتيب العناصر الثلاث كما في الشكل:

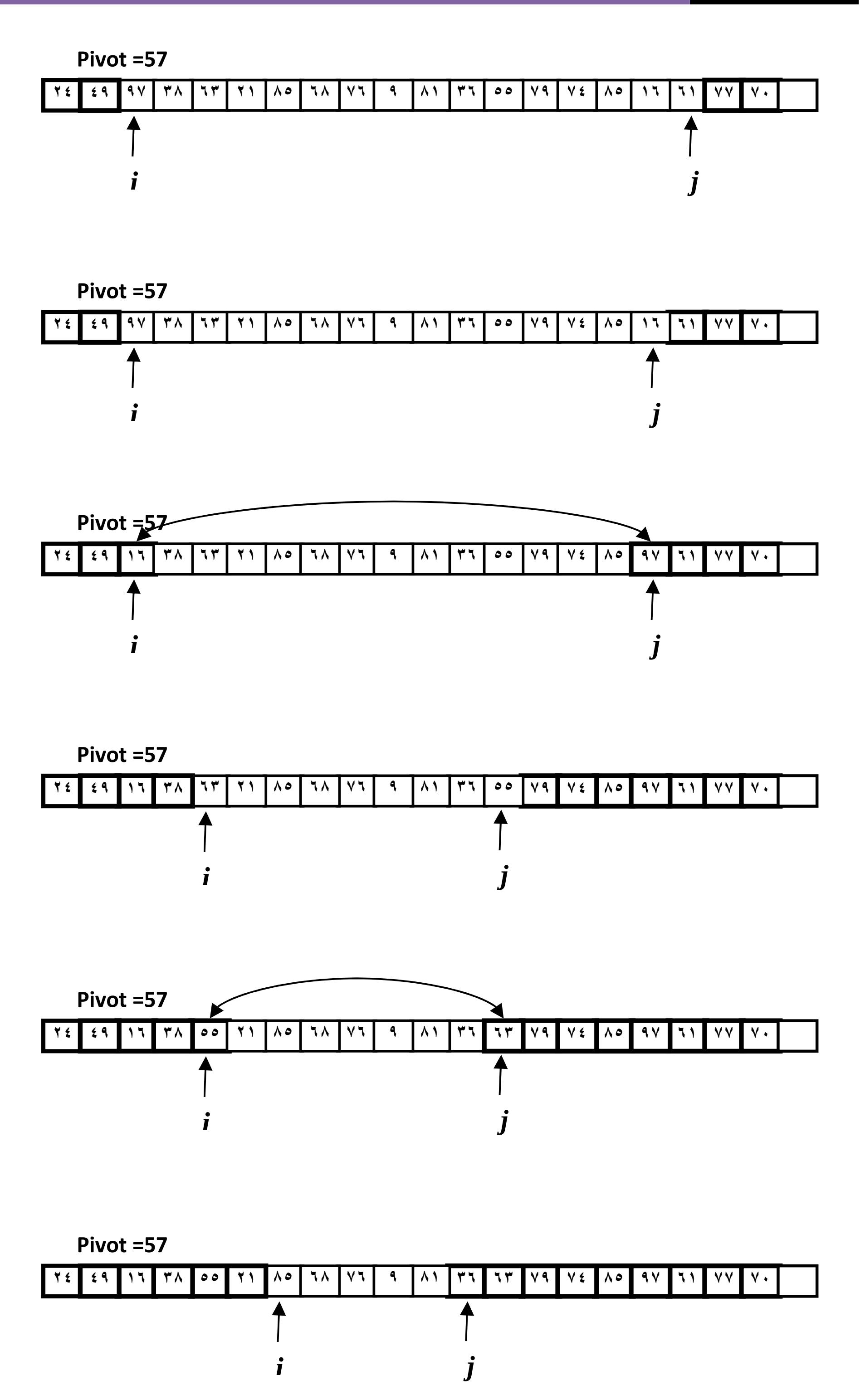


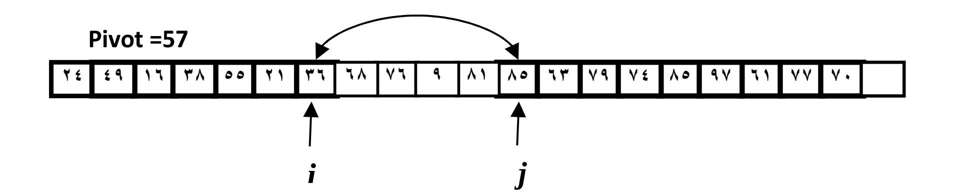
٣. نبدأ المقارنة من العنصر الثاني ، لان العنصر الأول هو الأصغر من الخطوة (٢).

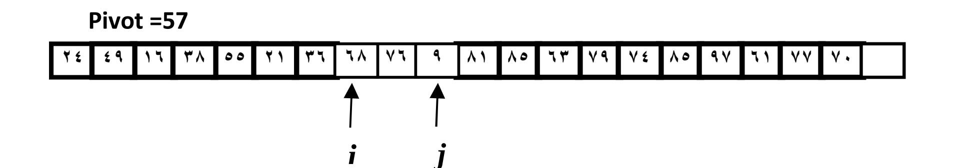


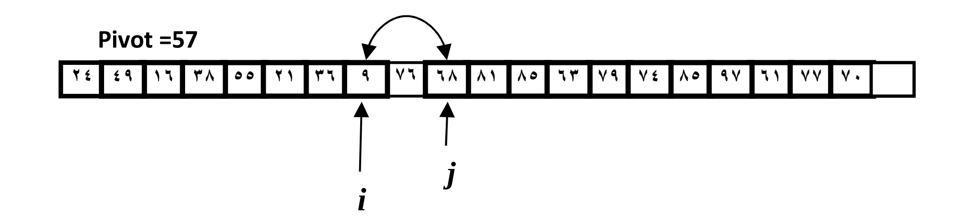


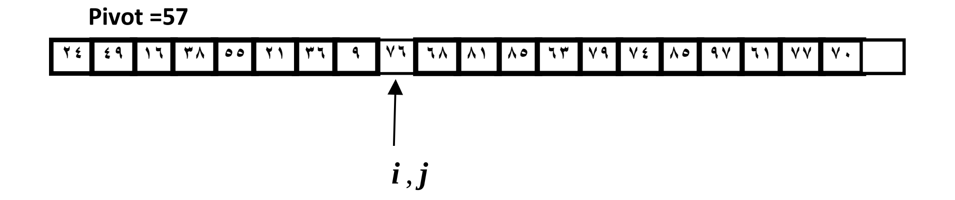


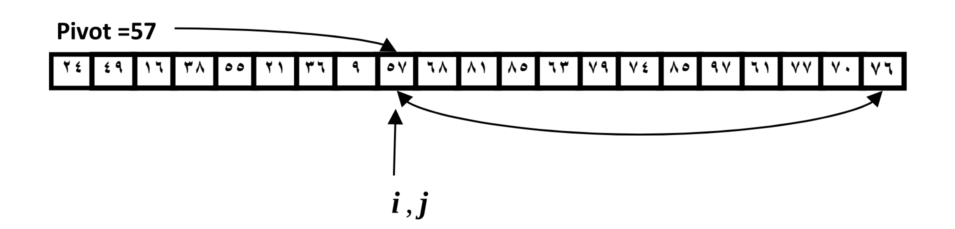

Pivot =57
i
j
Pivot =57
i
j
Pivot =57
i
j
Pivot =57
i
j
Pivot =57
i
j
Pivot =57
i
j





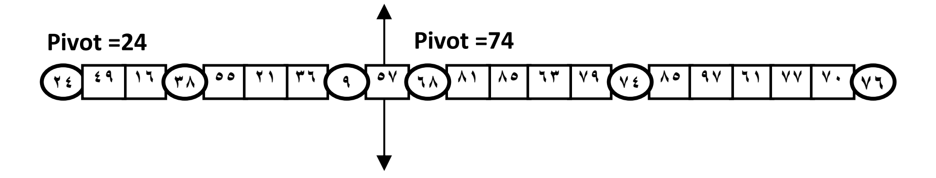






٤. بعد تحديد الموقع الصحيح للعنصر المنتخب ، نلاحظ أن جميع العناصر التي قيمها أصغر من العنصر المنتخب أصبحت في جهة اليسار وبقية العناصر في جهة اليمين.

٥. تطبيق الخوارزمية من جديد على كل جزء من القائمة وبنفس الخطوات المذكورة سابقاً كما في الشكل :



٦. نستمر بتطبيق الخوارزمية لحين الوصول إلى قائمة مرتبة وكما في الشكل التالي:

٩	١٦	٢١	٢٤	٣٦	٣٨	٤٩	٥٥	٥٧	٦١	٦٣	٦٨	٧٠	٧٤	٧٦	٧٧	٧٩	٨١	٨٥	٨٥	٩٧

Example:-

```
program quick1;
const maxlength=30;
type
list=array[1..maxlength] of integer;
var num:list;
first,last,length,index:integer;

procedure fillarray(var num:list;var length:integer);
var index:integer;
begin
writeln('enter the numbers of elements in the list:');
readln(length);
writeln('enter the element of the list:');
for index:=1 to length do
begin
write('num',index,'=');
readln(num[index]);
end;
end;

procedure quicksort(var num:list;left,right:integer);
var pivot,temp,leftarrow,rightarrow:integer;
begin
leftarrow:=left;
rightarrow:=right;
pivot:=num[(left+right) div 2];
repeat
```

```
while num[rightarrow]>pivot do
rightarrow:=rightarrow-1;
while num[leftarrow]<pivot do
leftarrow:=leftarrow+1;
if leftarrow<=rightarrow then
begin
temp:=num[leftarrow];
num[leftarrow]:=num[rightarrow];
num[rightarrow]:=temp;
leftarrow:=leftarrow+1;
rightarrow:=rightarrow-1;
end;
until rightarrow<leftarrow;
if left<rightarrow then
quicksort(num,left,rightarrow);
if leftarrow<right then
quicksort(num,leftarrow,right);
end;

procedure printlist(num:list;length:integer);
var index:integer;
begin
writeln;
writeln('the sorted list is:');
writeln;
for index:=1 to length do
write(' ',num[index]);
writeln;
end;

begin {main program}
fillarray(num,length);
writeln('the list befor sorting is:');
writeln;
for index:=1 to length do
write(' ',num[index]);
```

```
writeln;
quicksort(num,1,length);
printlist(num,length);
readln;
end.
```

Output:-

```
enter the numbers of elements in the list:
4
enter the element of the list:
num1=12
num2=5
num3=33
num4=2
the list befor sorting is:
 12 5 33 2
the sorted list is:
 2 5 12 33
```

Example:-

```
program quick2;
const m=4;
type
employee=record
name:string[20];
no:integer;
salary:real;
place:string[20];
scale:string[20];
```

```
end;
data=array[1..m]of employee;
var a:data; emp,min:employee;
loc,item,i,b,target,j,c,k,n,no:integer;
procedure linearsearch(item:integer);
begin
n:=m-1;
a[n+1].no:=item;
loc:=1;
while a[loc].no<>item do
loc:=loc+1;
if loc=n+1 then
writeln('this record is not found')
else
begin
writeln('the record location is by linear search is:',loc);
writeln('the data of this employee is:');
writeln('name       no       salary       place        scale');
with emp do
writeln(a[loc].name,'       ',a[loc].no,'       ',a[loc].salary,'
',a[loc].place,'       ',a[loc].scale);
end;
end;

procedure quicksort(var num:data;left,right:integer);
var leftarrow,rightarrow:integer;
temp,pivot:employee;
begin
leftarrow:=left;
rightarrow:=right;
pivot:=num[(left+right) div 2];
repeat
while num[rightarrow].no>pivot.no do
rightarrow:=rightarrow-1;
while num[leftarrow].no<pivot.no do
leftarrow:=leftarrow+1;
```

```
if leftarrow<=rightarrow then
begin
temp:=num[leftarrow];
num[leftarrow]:=num[rightarrow];
num[rightarrow]:=temp;
leftarrow:=leftarrow+1;
rightarrow:=rightarrow-1;
end;
until rightarrow<leftarrow;
if left<rightarrow then
quicksort(num,left,rightarrow);
if leftarrow<right then
quicksort(num,leftarrow,right);
end;


function binarysearch(var a:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1; last:=3;
repeat
midd:=(first+last) div 2;
if a[midd].no>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (a[midd].no=target);
if a[midd].no=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin {main program}
writeln('enter the number of emplyees:');
readln(no);
```

```
writeln('enter the data of employees:');
for i:=1 to no do
begin
write('enter the name of employee',i,': ');
readln(a[i].name);
write('enter the no of employee',i,': ');
readln(a[i].no);
write('enter the salary of employee',i,': ');
readln(a[i].salary);
write('enter the place of employee',i,': ');
readln(a[i].place);
write('enter the scale of employee',i,': ');
readln(a[i].scale);
end;
writeln('the data of employee are:');
writeln('name       no       salary        place         scale');
for i:=1 to no do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
writeln;
writeln('enter the no of employee to search by linear search:');
readln(item);
linearsearch(item);
writeln;
quicksort(a,1,no);
writeln('the sort of record is:');
writeln('name       no       salary        place         scale');
for i:=1 to no do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
writeln;
writeln('enter the no of employee to search by binary search:');
readln(target);
b:=binarysearch(a,target);
if a[b].no=target then
begin
```

```
writeln('the record location by binary search is=',b);
writeln('the data of this employee is:');
writeln('name       no       salary        place         scale');
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'       ',a[b].scale);
a[b].name:='0';
a[b].no:=0;
a[b].salary:=0;
a[b].place:='0';
a[b].scale:='0';
writeln;
writeln('name       no       salary        place         scale');
writeln(a[b].name,'       ',a[b].no,'       ',a[b].salary,'
',a[b].place,'      ',a[b].scale);
end
else
writeln('this record is not found');
writeln;
writeln('the data of employee are:');
writeln('name       no       salary        place         scale');
for i:=1 to no do
writeln(a[i].name,'       ',a[i].no,'       ',a[i].salary,'
',a[i].place,'       ',a[i].scale);
readln;
end.
```

Output:-

```
enter the number of emplyees:
2
enter the data of employees:
enter the name of employee1: duaa
enter the no of employee1: 2
enter the salary of employee1: 1132.3223
enter the place of employee1: sudia
enter the scale of employee1: q
```

```
enter the name of employee2: mariam

enter the no of employee2: 22

enter the salary of employee2: 323.231

enter the place of employee2: sudia

enter the scale of employee2: r

the data of employee are:

name       no       salary                place        scale

duaa       2        1.1323223000E+03      sudia        q

mariam     22       3.2323100000E+02      sudia        r


enter the no of employee to search by linear search:

2

the record location is by linear search is:1

the data of this employee is:

name       no       salary                place        scale

duaa       2        1.1323223000E+03      sudia        q


the sort of record is:

name       no       salary                place        scale

duaa       2        1.1323223000E+03      sudia        q

mariam      22        3.2323100000E+02      sudia      r


enter the no of employee to search by binary search:
2
the record location by binary search is=1
the data of this employee is:
name       no       salary                place        scale
duaa       2        1.1323223000E+03      sudia        q

name       no       salary                place        scale
0          0        0.0000000000E+00      0            0

the data of employee are:
name       no       salary                place        scale
0          0        0.0000000000E+00      0            0
```

```
mariam      22        3.2323100000E+02        sudia         r
```

(٩-2) خوارزميات الترتيب الخارجي External Sorting Algorithms :-

الدمج الثنائي Merg Sort :-

Example:-

```
program merge1؛
type
student=record
name:string؛
idno:integer؛
end؛
var st1,st2,st3:array[1..5] of student؛
i,j,c,k,n:integer; min:student؛
f1,f2,f3:file of student؛

procedure insertion_1؛
begin
for i:=1 to n do
begin
k:=0؛
min:=st1[i];
for j:=i to n do
if min.idno>st1[j].idno then
begin
min:=st1[j];
k:=j؛
end؛
```

```
for c:=k downto i do
st1[c]:=st1[c-1];
st1[i]:=min؛
end؛
end؛

procedure insertion_2؛
begin
for i:=1 to n do
begin
k:=0؛
min:=st2[i];
for j:=i to n do
if min.idno>st2[j].idno then
begin
min:=st2[j];
k:=j؛
end؛
for c:=k downto i do
st2[c]:=st2[c-1];
st2[i]:=min؛
end؛
end؛
procedure mergesort؛
begin
reset(f1);
reset(f2);
rewrite(f3);
read(f1,st1[i]);
```

```
read(f2,st2[i]);
repeat
if st1[i].idno<st2[i].idno then
begin
write(f3,st1[i]);
if not eof(f1) then
read(f1,st1[i]);
end
else
if st1[i].idno=st2[i].idno then
begin
write(f3,st1[i]);
if (not eof(f1)) and (not eof(f2)) then
begin
read(f1,st1[i]);
read(f2,st2[i]);
end؛
end
else
begin
write(f3,st2[i]);
if not eof(f2) then
read(f2,st2[i]);
end؛
until (eof(f1)) and (eof(f2));
if st1[i].idno<st2[i].idno then
begin
write(f3,st1[i]);
write(f3,st2[i]);
```

```
end
else
begin
write(f3,st2[i]);
write(f3,st1[i]);
end؛
close(f1);
close(f2);
close(f3);
end؛
begin {main program}
assign(f1,'c:\file1.dat’);
assign(f2,'c:\file2.dat’);
assign(f3,'c:\file3.dat’);
rewrite(f1);
rewrite(f2);
writeln('enter the value of n:’);
readln(n);
writeln('enter the data of file_1:’);
for i:=1 to n do
with st1[i] do
begin
write('enter the name[',i،’]:’);
readln(name);
write('enter the idno[',i،’]:’);
readln(idno);
end؛
insertion_1؛
for i:=1 to n do
```

```
write(f1,st1[i]);
close(f1);
writeln('the data in file_1 is:’);
reset(f1);
writeln('name    idno’);
while not eof(f1) do
begin
read(f1,st1[i]);
with st1[i] do
writeln(name,'    ',idno);
end؛
close(f1);
writeln('enter the data of file_2:’);
for i:=1 to n do
with st2[i] do
begin
write('enter the name[',i،’]:’);
readln(name);
write('enter the idno[',i،’]:’);
readln(idno);
end؛
insertion_2؛
for i:=1 to n do
write(f2,st2[i]);
close(f2);
writeln('the data in file_2 is:’);
reset(f2);
writeln('name    idno’);
while not eof(f2) do
```

```
begin
read(f2,st2[i]);
with st2[i] do
writeln(name,'    ',idno);
end;
close(f2);
mergesort;
writeln('the data in file_3 is:');
writeln;
reset(f3);
writeln('name    idno');
while not eof(f3) do
begin
read(f3,st3[i]);
with st3[i] do
writeln(name,'    ',idno);
end;
close(f3);
readln;
end.
```

Output:-

```
enter the value of n:
3
enter the data of file_1:
enter the name[1]:samy
enter the idno[1]:13
enter the name[2]:mohamed
enter the idno[2]:1
enter the name[3]:ahamd
```

```
enter the idno[3]:135
the data in file_1 is:
name      idno
mohamed   1
samy      13
ahamd     135
enter the data of file_2:
enter the name[1]:fatima
enter the idno[1]:16
enter the name[2]:yaser
enter the idno[2]:10
enter the name[3]:walla
enter the idno[3]:3
the data in file_2 is:
name     idno
walla    3
yaser    10
fatima    16
the data in file_3 is:
name      idno
mohamed   1
walla     3
yaser     10
samy      13
fatima    16
ahamd     135
```

Example:-

```
program merge2;
```

```
const m=5;
type
student=record
name:string[20];
no:integer;
salary:real;
place:string[20];
scale:string[20];
end;
data=array[1..m]of student;
var st1,st2,st3:data;
min:student;
loc,item,b,target,i,j,c,k,n,l:integer;
f1,f2,f3:file of student;

procedure linearsearch(item:integer);
begin
l:=m-1;
st3[l+1].no:=item;
loc:=1;
while st3[loc].no<>item do
loc:=loc+1;
if loc=n+1 then
writeln('this record is not found')
else
begin
writeln('the record location is:',loc);
writeln('the data of this student is:');
writeln('name       no       salary        place         scale');
```

```
with st3[loc] do

writeln(name,'        ',no,'        ',salary,'        ',place,'
',scale);

end;

end;


procedure insertion_1;

begin

for i:=1 to n do

begin

k:=0;

min:=st1[i];

for j:=i to n do

if min.no>st1[j].no then

begin

min:=st1[j];

k:=j;

end;

for c:=k downto i do

st1[c]:=st1[c-1];

st1[i]:=min;

end;

end;


procedure insertion_2;

begin

for i:=1 to n do

begin

k:=0;

min:=st2[i];
```

```
for j:=i to n do
if min.no>st2[j].no then
begin
min:=st2[j];
k:=j;
end;
for c:=k downto i do
st2[c]:=st2[c-1];
st2[i]:=min;
end;
end;

procedure mergesort;
begin
reset(f1);
reset(f2);
rewrite(f3);
read(f1,st1[i]);
read(f2,st2[i]);
repeat
if st1[i].no<st2[i].no then
begin
write(f3,st1[i]);
if not eof(f1) then
read(f1,st1[i]);
end
else
if st1[i].no=st2[i].no then
begin
```

```
write(f3,st1[i]);
if (not eof(f1)) and (not eof(f2)) then
begin
read(f1,st1[i]);
read(f2,st2[i]);
end;
end
else
begin
write(f3,st2[i]);
if not eof(f2) then
read(f2,st2[i]);
end;
until (eof(f1)) and (eof(f2));
if st1[i].no<st2[i].no then
begin
write(f3,st1[i]);
write(f3,st2[i]);
end
else
begin
write(f3,st2[i]);
write(f3,st1[i]);
end;
close(f1);
close(f2);
close(f3);
end;
```

```
function binarysearch(var st3:data;target:integer):integer;
var first,last,midd:integer;
begin
first:=1; last:=4;
repeat
midd:=(first+last) div 2;
if st3[midd].no>target then
last:=midd-1
else
first:=midd+1;
until (first>last) or (st3[midd].no=target);
if st3[midd].no=target then
binarysearch:=midd
else
binarysearch:=0;
end;

begin {main program}
assign(f1,'c:\tp\file1.dat');
assign(f2,'c:\tp\file2.dat');
assign(f3,'c:\tp\file3.dat');
rewrite(f1);
rewrite(f2);
writeln('enter the number of student:');
readln(n);
writeln('enter the data of file_1:');
for i:=1 to n do
with st1[i] do
```

```
begin
writeln('enter the data of student[',i,']:');
write('enter the name: ');
readln(name);
write('enter the no: ');
readln(no);
write('enter the salary: ');
readln(salary);
write('enter the place: ');
readln(place);
write('enter the scale: ');
readln(scale);
end;
insertion_1;
for i:=1 to n do
write(f1,st1[i]);
close(f1);
writeln('the data in file_1 is:');
reset(f1);
writeln('name       no       salary        place         scale');
while not eof(f1) do
begin
read(f1,st1[i]);
with st1[i] do
writeln(name,'       ',no,'       ',salary,'       ',place,'
',scale);
writeln;
end;
close(f1);
```

```
writeln('enter the data of file_2:');
for i:=1 to n do
with st2[i] do
begin
writeln('enter the data of student[',i,']:');
write('enter the name: ');
readln(name);
write('enter the no: ');
readln(no);
write('enter the salary: ');
readln(salary);
write('enter the place: ');
readln(place);
write('enter the scale: ');
readln(scale);
end;
insertion_2;
for i:=1 to n do
write(f2,st2[i]);
close(f2);
writeln('the data in file_2 is:');
reset(f2);
writeln('name       no       salary        place         scale');
while not eof(f2) do
begin
read(f2,st2[i]);
with st2[i] do
writeln(name,'       ',no,'       ',salary,'       ',place,'
',scale);
writeln;
```

```
end;
close(f2);
mergesort;
writeln('the data in file_3 is:');
writeln;
reset(f3);
writeln('name       no       salary        place         scale');
while not eof(f3) do
begin
read(f3,st3[i]);
with st3[i] do
writeln(name,'       ',no,'       ',salary,'       ',place,'
',scale);
writeln;
end;
close(f3);
reset(f3);
i:=1;
while not eof(f3) do
begin
read(f3,st3[i]);
i:=i+1;
end;
close(f3);

writeln('enter the no of student to search-->linear search:');
readln(item);
linearsearch(item);
writeln;
writeln('enter the no of student to search-->binary search:');
```

```
readln(target);

b:=binarysearch(st3,target);

if st3[b].no=target then

begin

writeln('the record location=',b);

writeln('the data of this student is:');

writeln('name       no       salary        place         scale');

with st3[b] do

begin

writeln(name,'       ',no,'       ',salary,'       ',place,'
',scale);

name:='0';

no:=0;

salary:=0;

place:='0';

scale:='0';

end;

writeln;

writeln('name       no       salary        place         scale');

with st3[b] do

writeln(name,'       ',no,'       ',salary,'       ',place,'
',scale);

end

else

writeln('this record is not found');

writeln;

writeln('the data of student are:');

writeln('name       no       salary        place         scale');

for i:=1 to n do

with st3[i] do
```

```
writeln(name,'        ',no,'        ',salary,'        ',place,'
',scale);

readln;

end.
```

Output:-

```
enter the number of student:

2

enter the data of file_1:

enter the data of student[1]:

enter the name: wegdo

enter the no: 23

enter the salary: 123.434

enter the place: sudan

enter the scale: w

enter the data of student[2]:

enter the name: lubna

enter the no: 5

enter the salary: 1232.323

enter the place: sudan

enter the scale: e

the data in file_1 is:

name        no       salary                place         scale

lubna       5        1.2323230000E+03      sudan         e

wegdo       23       1.2343400000E+02      sudan         w


enter the data of file_2:

enter the data of student[1]:

enter the name: fatima

enter the no: 8
```

```
enter the salary: 1323.232

enter the place: sudan

enter the scale: q

enter the data of student[2]:

enter the name: haneen

enter the no: 97

enter the salary: 23.3232

enter the place: sudan

enter the scale: y

the data in file_2 is:

name         no        salary                  place          scale

fatima       8         1.3232320000E+03        sudan          q

haneen       97        2.3323200000E+01        sudan          y


the data in file_3 is:


name         no        salary                  place          scale

lubna        5         1.2323230000E+03        sudan          e

fatima       8         1.3232320000E+03        sudan          q

wegdo        23        1.2343400000E+02        sudan          w

haneen       97        2.3323200000E+01        sudan          y


enter the no of student to search-->linear search:

8

the record location is:2

the data of this student is:

name         no        salary                  place          scale

fatima       8         1.3232320000E+03        sudan          q
```

```
enter the no of student to search-->binary search:
5
the record location=1
the data of this student is:
name       no      salary              place        scale
lubna      5       1.2323230000E+03    sudan        e

name       no      salary              place        scale
0          0       0.0000000000E+00    0            0

the data of student are:
name       no      salary              place        scale
0          0       0.0000000000E+00    0            0
fatima     8       1.3232320000E+03    sudan        q
```

****

الوحدة العاشره

الأشجار

Trees

(١-١٠) مقدمة عن الأشجار كبنية بيانات :-

إن بنية البيانات هذه لها درجة عالية من الأهمية ، وهي تحتوي على بيانات ترتبط فيما بينها عن طريق فروع Branches ، أي أنها لاتعتمد على التركيب الخطي في تكوينها ، فليس بالضرورة أن يرتبط العنصر فيها بعنصر واحد أمامه وعنصر واحد خلفه .

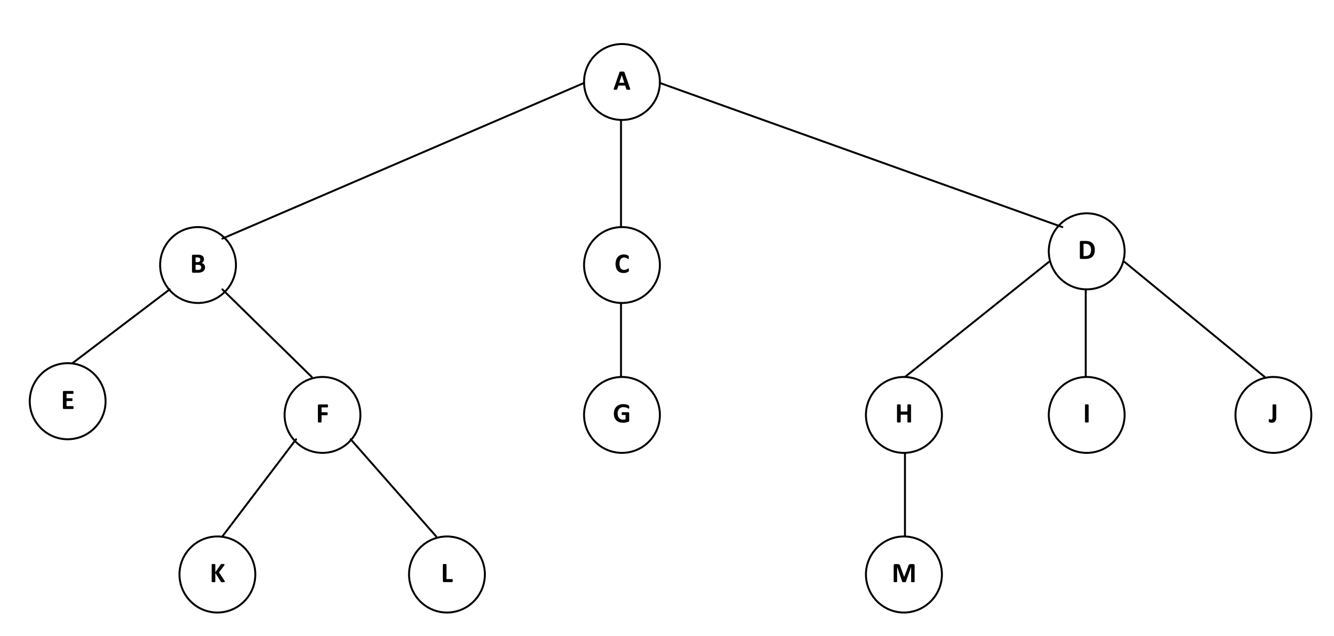
تعريف :-

الشجرة هي مجموعة محدودة ، تتكون من عقدة واحده أو أكثر بحيث :-

١- توجد بها عقدة مصممة بشكل خاص تسمى الجذر root .

٢- العقد الباقيه مقسمة إلى N>=0 من المجموعات T1.....Tn المنفصله بحيث كل واحده من هذه المجموعات تكون شجرة T1.....Tn تسمى شجريرات Subtrees من الجذر .

أنظر للشجرة التالية وحاول أن تربط بين ماسبق من التعريفات وماهو موضح بالشكل لتستطيع أن تتخيل شكل الشجرة :-



- إن الشجرة السابقة بها ١٣ عقدة node ، يحتوي عنصر البيانات في كل عقدة على حرف ، جذر هذه الشجرة هي العقدة التي يحتوي عنصر البيانات فيها على الحرف A ، لاحظ أننا رسمنا الشجرة وجذرها إلى الأعلى ، وهذا هو المنهج المتبع في رسم الأشجار ، إن عدد الأشجار التي تتفرع من عقدة ما يسمى درجة العقدة Degree of node فدرجة العقدة التي تحتوي على على A هي ٣ بينما درجة العقدة التي تحتوي على C هي ١ ، أما درجة العقدة K فهي صفر ، إن العقدة التي تكون درجتها مساوية للصفر تسمى ورقة Leaf أو عقدة نهائية Terminal node ، بناء على ماسبق فإن {K,L,G,M,I,J} هي مجموعة العقد الورقية ، وبعكس ذلك فإن العقد التي درجتها لا تساوي الصفر تسمى عقداً داخليه Nonterminal Node ماعدا الجذر ، إن جذور الشجيرات التي تتفرع من عقدة ما ولتكن X تسمى أبناء Children لـ X ، و X يسمى أباً Parent لهؤلاء الأبناء ، وبالنظر إلى شجرتنا السابقه نلاحظ أن أبناء D هم H,I,J ، وأب D هو A .

- إن بناء الأب الواحد يسمون أشقاء Siblings ، وبذلك فإن H,I,J هم أشقاء لأن لهم نفس الأب D .

درجة الشجرة Degree of tree :
هي أعلى درجة للعقد في الشجرة ، وبالتالي فإن درجة الشجرة السابقة هي ٣ .

مستوى العقدة Level of tree :
إذا قلنا أن عقدة ما في المستوى L ، فإن أبنائها سوف يكونون في المستوى L+1 ، فإذا افترضنا أن الجذر يقع في المستوى ١ فإن الإبن d يقع في المستوى ٢ .

إرتفاع الشجرة Height of tree :
هو أعلى مستوى يمكن أن تأخذها عقدة في الشجرة ، وبذلك يكون إرتفاع الشجرة السابقة هو ٣ .

الغابة Forest :
هي مجموعة بها عدد n>=0 من الأشجار المنفصلة عن بعضها . لا حظ أننا إذا أزلنا جذر الشجرة فإننا سوف نحصل على غابة ، على سبيل المثال في الشجرة السابقة إذا أزلنا A فسوف نحصل على غابة مكونة من ٣ أشجار .

(١٠-٢) أنواع الأشجار:-

١- الأشجار الثنائية .
٢- الأشجار المتوازنه .

(١٠-٣) الأشجار الثنائية Binary Tree :-

تعتبر الأشجار الثنائية من الأنواع المهمة في بنيات الأشجار ، وهي تتصف بأن أي عقدة فيها يمكن أن يكون لديها على الأكثر فرعان ، أي أنه لاتوجد بها عقدة درجتها أكبر من ٢ ، في الأشجار الثنائية فإننا نميز بين الشجيرة اليمنى والشجيرة اليسرى للعقدة الواحده .

تعريف:-

الشجرة الثنائية هي مجموعة محدودة من العقد ، وهذه المجموعة إما أن تكون خالية ، أو أنها تحتوي على جذر وشجيرتين على الأكثر منفصلتين تسميان الشجيرة اليسرى Left Subtree ، والشجيرة اليمنى Right Subtree ، وهاتان الشجرتان متصلتان بالجذر ، وكل منهما تمثل شجرة .

لاحظ أن هذا التعريف هو تعريف مجرد ، ولتحديد الشجرة الثنائية كنوع من أنواع البيانات المجردة ، يجب علينا أن نوضح ماهي العمليات التي يمكن أن تجرى على الشجرة الثنائية ، لاحظ أيضاً أن هذا التعريف لم يبين لنا طريقة ترتيب العقد داخل الشجرة .

إن الأشجار الثنائية تستخدم لعدة أغراض من ضمنها البحث ، وفي الفقرة التالية نستعرض نوع من أنواع الأشجار الثنائية التي تستخدم في عمليات البحث والترتيب .

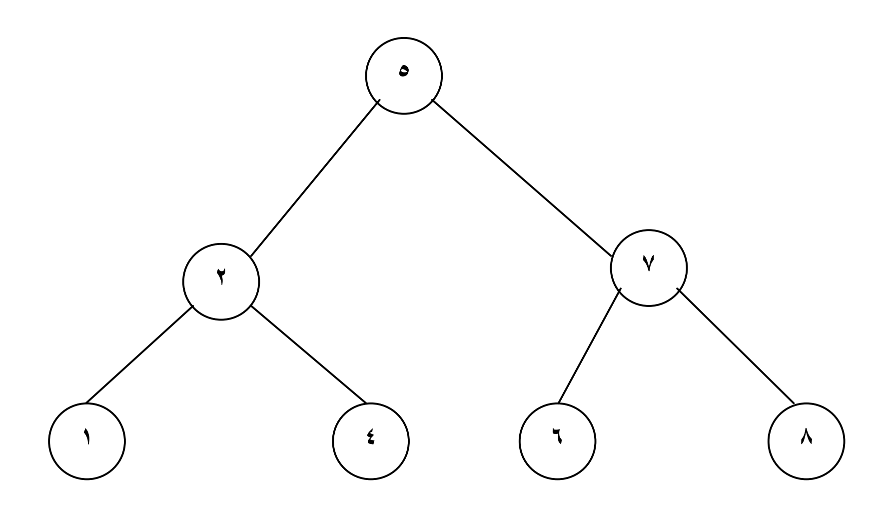
شجرة البحث الثنائيه Binary Search Tree :-

تعريف:-

هي شجرة ثنائية ، والتي إما أن تكون خالية أو أن كل عقدة فيها تحتوي على مفتاح يحقق الشروط التالية :

١- كل المفاتيح (إن وجدت) في الشجيرة اليسرى ، أصغر من مفتاح العقدة في الجذر.
٢- مفتاح العقدة الجذر أصغر من كل المفاتيح (إن وجدت) في الشجيرة اليمنى .
٣- الشجيرة اليمنى والشجيرة اليسرى تنطبق عليهما نفس الشروط السابقه.

والمثال التالي يوضح شكل شجرة البحث الثنائية :-



لاحظ أن كل المفاتيح التي تقع في الشجرة اليمنى المتصله بالجذر ٥ هي أكبر من ٥ ، وأن كل المفاتيح التي تقع في الشجيرة اليسرى المتصلة بالجذر ٥ هي أصغر من ٥ ، وكذلك لو أخذنا كل شجيرة من الشجيرتين T(2) و T(7) فإننا سوف نلاحظ نفس الملاحظة .

تمثيل شجرة البحث الثنائية في لغة باسكال :-

- لتسهيل التعامل مع الشجرة الثنائية فإننا سوف نستخدم بنية بيانات متحركه لتمثيلها ، سوف نستخدم المتغير مؤشر ليدلنا على مكان الشجرة . إن إسم الإسم المعتاد لذلك المؤشر هو root لأنه سوف يشير إلى جذر الشجرة . إذا كانت الشجرة خاليه فإن root=nil .
- كل عقدة في شجرة البحث الثنائية لها شجيرة يمنى وشجيرة يسرى ، وهذا مايمكن الحصول عليه بإستعمال المؤشرات في هذا الجزء من البرنامج .

```
program binary_search_tree;
type
 p=^node;
 node=record
   info:integer;
   left,right:p;
end;
var root,pt,pt1,c,s:p;
     r,z,x,k:integer;
     found:boolean;
```

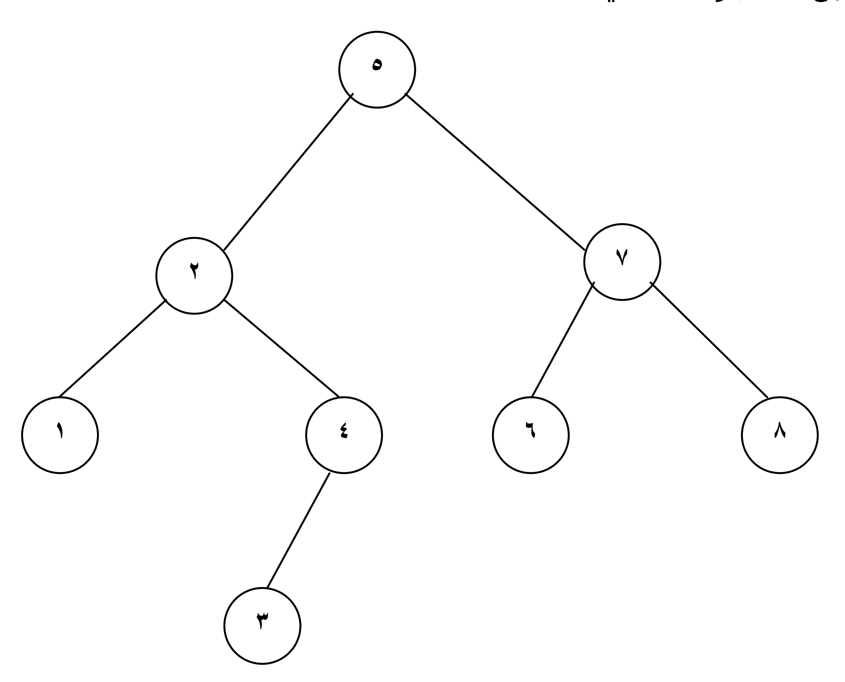
إنشاء عقدة في شجرة :-

الدالة التالية توضح لنا كيفية إنشاء عقدة تمهيداً لإضافتها إلى شجرة البحث الثنائية ، حيث يتم إنشاء مؤشر بإستخدام الدالة new ، والذي يشير إلى سجل يحتوي على ثلاثة حقول ، إثنين منهما عبارة عن مؤشرات ، والثالث من النوع integer ليخزن بداخله المفتاح .

```
function creatptr(r:integer):p;
var ptr:p;
begin
     new(ptr);
   with ptr^ do
begin
   info:=r;
   left:=nil;
   right:=nil;
end;
   creatptr:=ptr;
end;
```

إضافة عقدة إلى شجرة البحث الثنائية :-

لنفرض أننا نريد أن نضيف عقدة إلى شجرة البحث الثنائية التي في المثال السابق ، هذه العقدة تحتوي على المفتاح ٣ ، فنبدأ أولاً بمقارنة ٣ مع مفتاح الجذر ٥ نجد أن 5>3 ، الآن ننقل إلى الشجيرة اليسرى للجذر ، نلاحظ أن جذرها يحتوي على المفتاح ٢ ، والذي هو أصغر من ٣ ، لذلك نتجه الآن إلى الشجيرة اليمنى ، نجد أن المفتاح قيمته ٤ ، وهو أكبر من ٣ ، ولأن المؤشر الأيسر لهذه العقدة يشير إلى nil ، يمكننا إضافة العقدة الجديدة ، والتي تحتوي على المفتاح ٣ في هذا المكان ، لتصبح الشجرة كالتالي:



شكل(**)

البرنامج التالي يستخدم النداء الذاتي لإضافة العقدة المشار إليها بالمؤشر pt1 إلى الشجرة المشار إلى جذرها بالمؤشر pt :

```
procedure add(var pt,pt1:p);
begin
   if pt=nil then
begin
   pt:=pt1;
   pt^.left:=nil;
   pt^.right:=nil;
```

```
end
 else
   if pt1^.info<pt^.info then
   add(pt^.left,pt1)
 else
   add(pt^.right,pt1);
end;
```

البحث عن مفتاح في شجرة البحث الثنائية :-

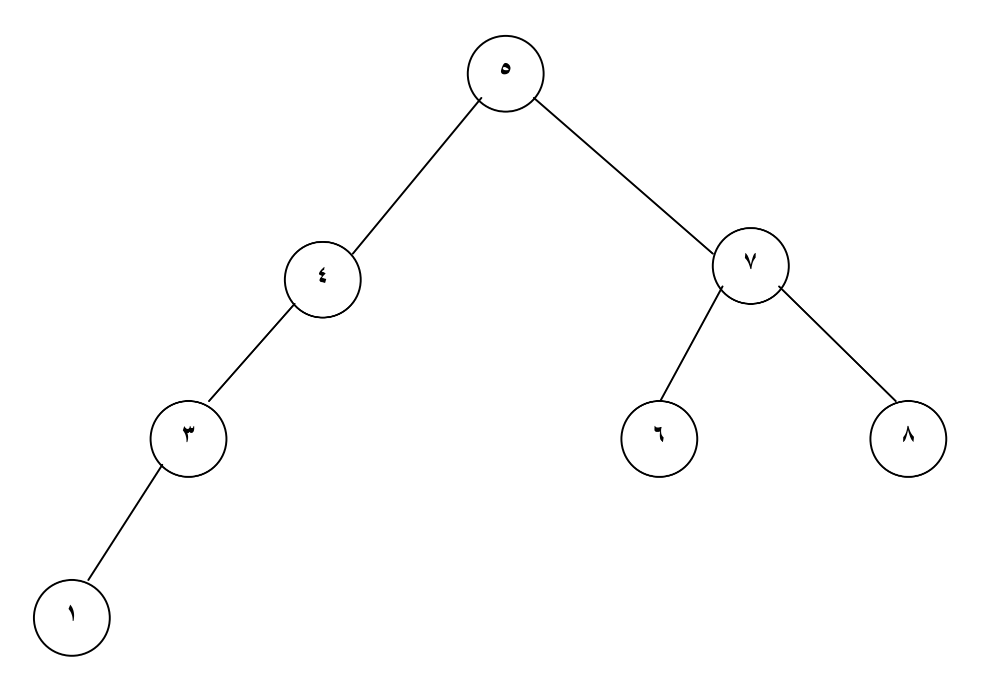
البرنامج التالي يوضح طريقة إستخدام النداء التكراري في البحث عن مفتاح Z في شجرة بحث ثنائية مشار إلى جذرها بالمؤشر P ، وهو يرجع لنا مؤشر يشير إلى العقدة التي تحتوي على المفتاح في حالة وجوده ويؤشر إلى nil في حالة عدم وجود المفتاح .

```
function search(pt:p; z:integer):p;
begin
   if pt=nil then
begin
   writeln(z,' is not found');
   search:=nil;
end
 else
   if pt^.info=z then
   search:=pt
 else
   if z<pt^.info then
   search:=search(pt^.left,z)
 else
   search:=search(pt^.right,z);
end;
```

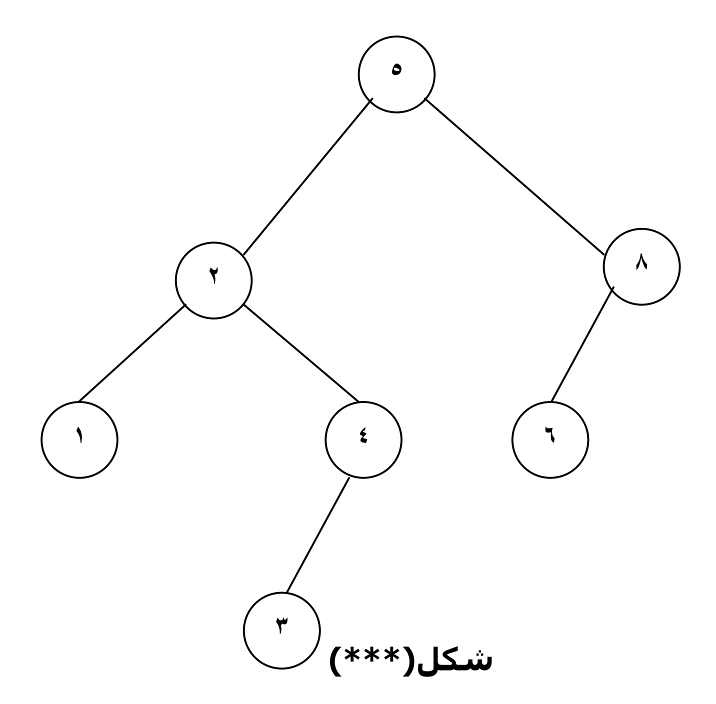
حذف عقدة من شجرة البحث الثنائية :-

إذا كانت العقدة المراد حذفها من الشجرة عبارة عن ورقة فإن الأمر في غاية السهولة كما هو موضح في الرسم ، حيث تم حذف العقدة التي تحتوي على المفتاح (٣) أما إذا كانت هذه العقدة لديها شجيرتان يمنى ويسرى فإن الأمر يبدو معقداً .

في الشكل(**) لنفرض أننا نريد حذف العقدة التي تحتوي على المفتاح ٢ ، فكل الذي علينا عمله هو أن نرفع الشجيرة اليسرى مكان العقدة ٢ ، ونجعل العقدة اليسار للعقدة المراد حذفها (٢) تتصل بالعقدة التي تقع أقصى اليمين السفلي للشجيرة ٤ ، ويصبح الشكل كالتالي :



الآن لنفرض أننا نريد حذف العقدة (٧) من الشكل(**) ، فالذي يجب عمله هو أن نرفع العقدة ٨ (العقده اليمنى) إلى أعلى وتكون العقده ٦ عبارة عن شجيرة يسرى للمفتاح (٨) كما هو موضح في الشكل (***) :



شكل(***)

والبرنامج التالي يوضح طريقة الحذف :-

```
procedure delet(root:p; var found:boolean; k:longint);
var temp:p;
begin
   if root^.info=k then
   found:=true
 else
   if root^.info>k then
   delet(root^.left,found,k)
 else
   delet(root^.right,found,k);
   if (found) and (k<>root^.info) then
begin
   if root^.left^.info=k then
begin
   temp:=root^.left;
   root^.left:=temp^.left;
end
 else
begin
   temp:=root^.right;
   root^.right:=temp^.left;
end;
   if temp^.right<>nil then
   add(root,temp^.right);
   found:=false;
   dispose(temp);
end;
end;
```

والبرنامج التالي للطباعة :-

```
procedure print(root:p);
begin
   while root<>nil do
begin
   write(root^.info,' ');
   root:=root^.left;
   write(root^.info,' ');
   root:=root^.right;
   write(root^.info,' ');
   writeln;
end;
end;
```

مناداة البرامج السابقه داخل البرنامج الرئيسي :-

```
begin {main program}

   writeln('1-->creatptr & add');
   writeln('2-->searsh');
   writeln('3-->delet');
   writeln('4-->print');
   writeln('5-->exit');
 repeat

   writeln;
   writeln('What Do You Want To Do..?');
   readln(x);
 case x of
1:begin
   writeln('Enter The Number You Want To Add:');
   readln(r);
   root:=creatptr(r);
   add(root,pt1);
end;
2:begin
   writeln('Enter The Number You Want To Search:');
   readln(z);
```

```
   s:=search(pt,z);
end;
3:begin
   writeln('Enter The Number You Want To Delet:');
   readln(k);
   delet(root,found,k);
end;
4:print(root);
end;
 until x=5;
  writeln('The End..');
Readln;
End.
```

(٤-١٠) الأشجار المتوازنه B-Tree :-

مقدمة عن الشجرة المتوازنة Balanced Tree:-

تعريف:-

هي عبارة عن شجرة متزنة وكل عقدة تحتوي علي مجموعة من المفاتيح الـ Keys والمؤشرات الـ Pointers وهي عبارة عن Dynamic ، ويتضح ذلك في عمليتي الإضافة والحدف، فالمستويات تقل وتزيد وترتب حسب العناصر التي تحتويها، وتستخدم في عملية البحث طريقة البحث الثنائي.

الشجرة المتوازنة من الدرجة T هي شجرة لها عقدة (جذر) وتتميز بالصفات التالية:-

١- كل عقدة في هذه الشجرة لديها الآتي:
ـ X[n] وهو عبارة عن عدد المفاتيح المخزنة في العقدة X .
ـ المفاتيح في العقدة X مرتبة ترتيبا تصاعديا أي أن :
K(1) <= K(2) <= K(3) <= .... <= K(n).
ـ العقدة في الشجرة المتوازنة تسمي ورقة Leaf إذا كان ليس لديها أبناء Children وإلا فإنها تسمي عقدة داخلية Internal Node .
٢- إداكانت X عقدة داخلية فإنها تحتوي علي N[x]+1 من المؤشرات تشير كل منها إلي ابن من أبنائها ، أما العقدة الورقية فليس لها أبناء ، لذلك فإن مؤشراتها تشير إلي اللاشيء Nil.
٣- كل الأوراق لها نفس الإرتفاع، والذي هو نفس ارتفاع الشجرة .
٤- هنالك حد أدني وحد أقصي لعدد المفاتيح التي يمكن أن تخزن في أي عقدة في الشجرة وتحكم هذه الحدود بإستعمال المتغير T>2 أي أن 2 هي أصغر درجة لشجرة متوازنة.

- ✓ كل العقد ما عدا عقدة الجذر تحتوي علي T-1 من المفاتيح كحد أدني، وبالتالي أي عقدة داخلية ماعدا عقدة الجذر تحتوي علي T من الأبناء كحد أدني أما عقدة الجذر فإنها يجب أن تحتوي علي مفتاح واحد كحد أدني.
- ✓ كل عقدة يمكن أن تحتوي علي عدد 2T-1 من المفاتيح كحد أقصي وبالتالي أي عقدة داخلية يمكن أن تحتوي علي عدد 2T من الأبناء كحد أقصي، لذلك فإن العقدة توصف بأنها ممتلئة إذا احتوت علي عدد 2T-1 من المفاتيح.

إستعمالات الـ B-Tree :-

تستخدم الـ B-Tree لتوضيح مفهوم الفهرسة في الملفات المفهرسة حيث إن الفهرسة الـ (Indexing) ماهي إلا B-Tree حيث إن كلاً منها يحتوي علي جدول به حقلين الأول يعطي الحقل المفتاحي الغير متكرر، والحقل الثاني يشير إلي السجل الذي يحمل هذه القيمة المفتاحية.
إن عملية قراءة المعلومات من ذاكرة تخزينٍ ثانويةٍ دائما ما تستهلك وقتا كبيرا إذا ما قورن بالوقت الذي تستغرقه العمليات التي تجرى علي هذه المعلومات المسترجعة من الذاكرة الثانوية، وهذه الظاهرة تكون واضحة خاصة في معالجة قاعدة بيانات ضخمة، وقراءة صفحات من المعلومات ونقلها إلي الذاكرة الرئيسية في الحاسب تحتاج إلي أن يقوم الحاسب بالبحث عدة مرات عن الصفحة المحددة في الذاكرة الثانوية ممايستلزم استهلاك وقت كبير.

يمكن إستعمال الشجرة المتوازنة لتخزين هذه الصفحات للقراءة والتعديل ومن ثم إعادتها إلي الذاكرة الثانوية مرة أخري.
إن كل عقدة في الشجرة المتوازنة يمكن أن تحتوي علي أكثر من صفحة وكل ماكان عامل التفرع كبيرا كان ارتفاع الشجرة أقل وبالتالي يصغر زمن البحث عن هذه الصفحة.

الفرق بين الـ Balanced Tree و Binary Tree :-

يحتوي الـ B-Tree علي جميع العناصر أوالمفاتيح في المستوي الأخير(الورقي)، بينما الـ Binary Tree لايحتوي علي جميع العناصر في المستوي الأخير.
عمليات على الشجرة المتوازية:Balance tree operation's

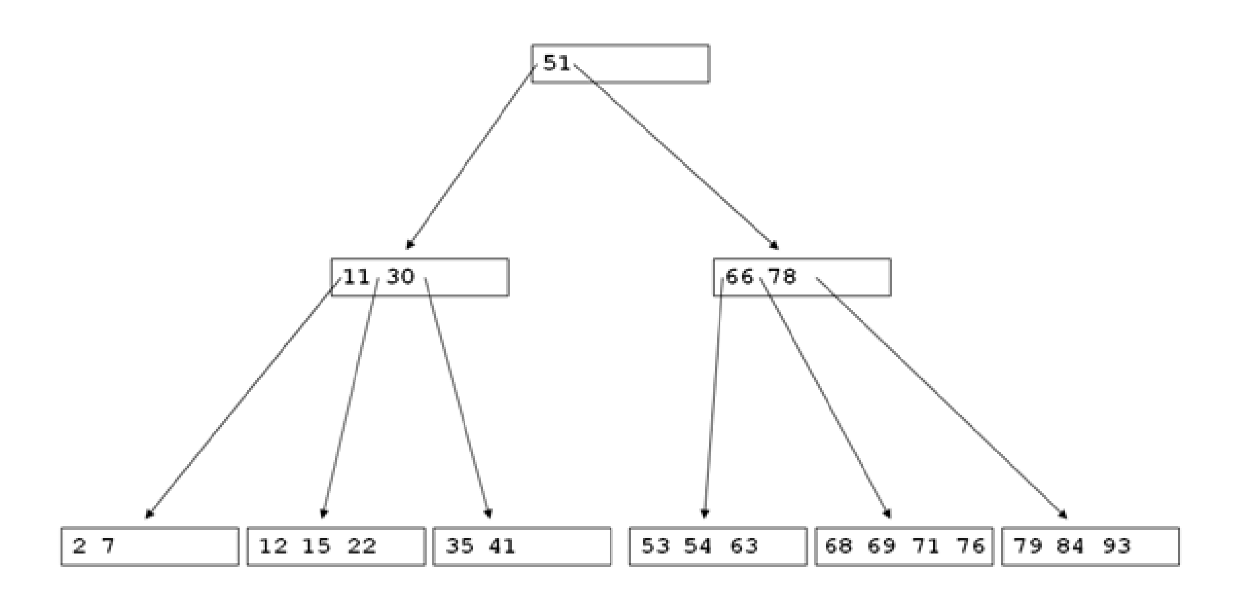
أولاً: عملية البحث

خوارزمية البحث:-

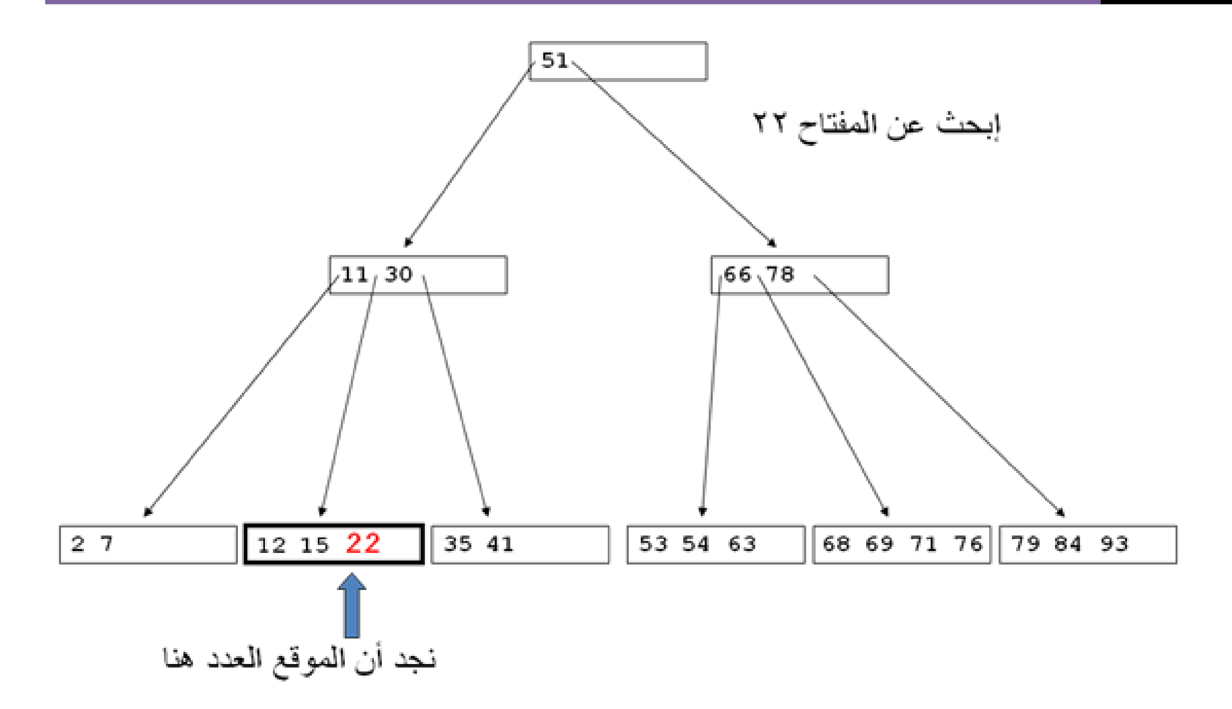
إبدأ من الجذر ، إستخدم مفتاح البحث للوصول إلى الورق، إذا كان مفتاح البحث مساوي للجذر إذا تم إيجاد السجل وإلا إذا كان أقل من الجذر إذهب المؤشر اليسار وإلا إذا كان أكبر من الجذر إذهب إلى مؤشر اليمين.

Search: Start at root; use key comparisons to go to leaf.
If search key is equal to root node then record is found else
If its less than root node go to the left pointer else if greater than go to the right pointer.

مثال:
إذا كان لدينا الشجرة المتوازية الاتية :-



وإذا أردنا البحث عن العدد ٢٢ نجد أن :



ثانيا: عملية الإضافة: insert operation

خوارزمية الإضافة:-

١- أوجد الموقع الصحيح للمفتاح (بإستخدام عملية الإضافة).
٢- أضف المفتاح المراد إضافتة إلى العقدة الصحيحة.
٣- إذا كان هناك مساحة كافة ، تمت العملية وإلا إذهب إلى الخطوة التالية.
٤- إقسم العقدة إلى عقدتين
أعد توزيع المداخل بإنتظام بين العقدة الحالية والعقدة الجديدة.
٥- أضف العنصر الوسط إلى عقدة الأب وأضف العنصرين الذين يلي العنصر الوسط إلى العقدة الجديدة.
٦- إذهب إلى الخطوة رقم ٣.

1) Find correct leaf node.

2) Add index entry to the node.

3) If enough space, *done*!.

4) Else, *split* the node.

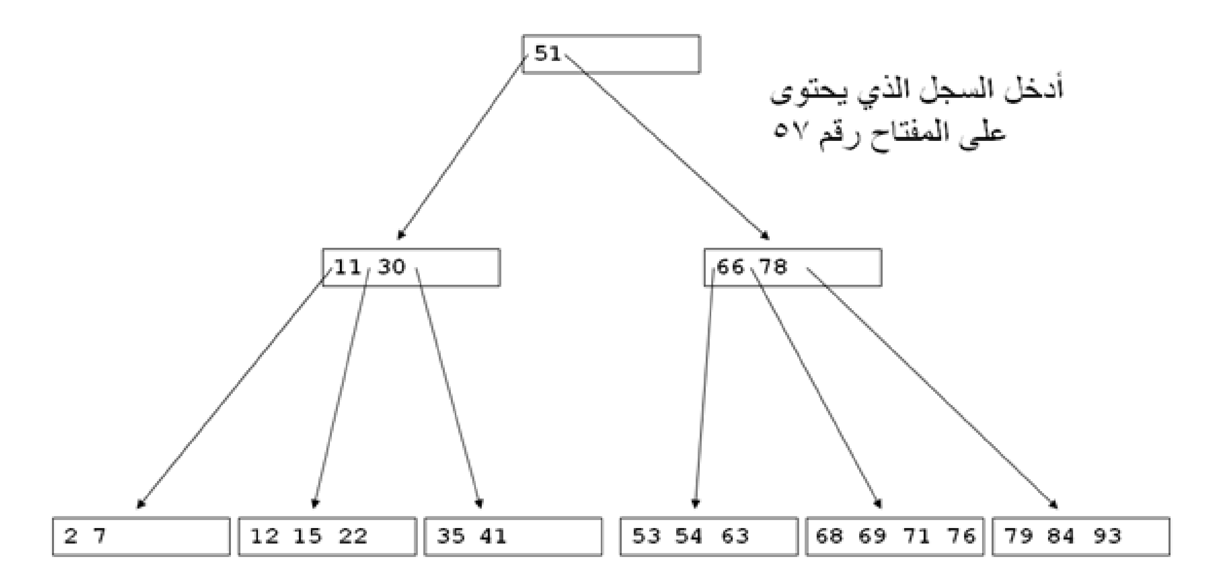
Redistribute entries evenly between the current.

node and the new node.

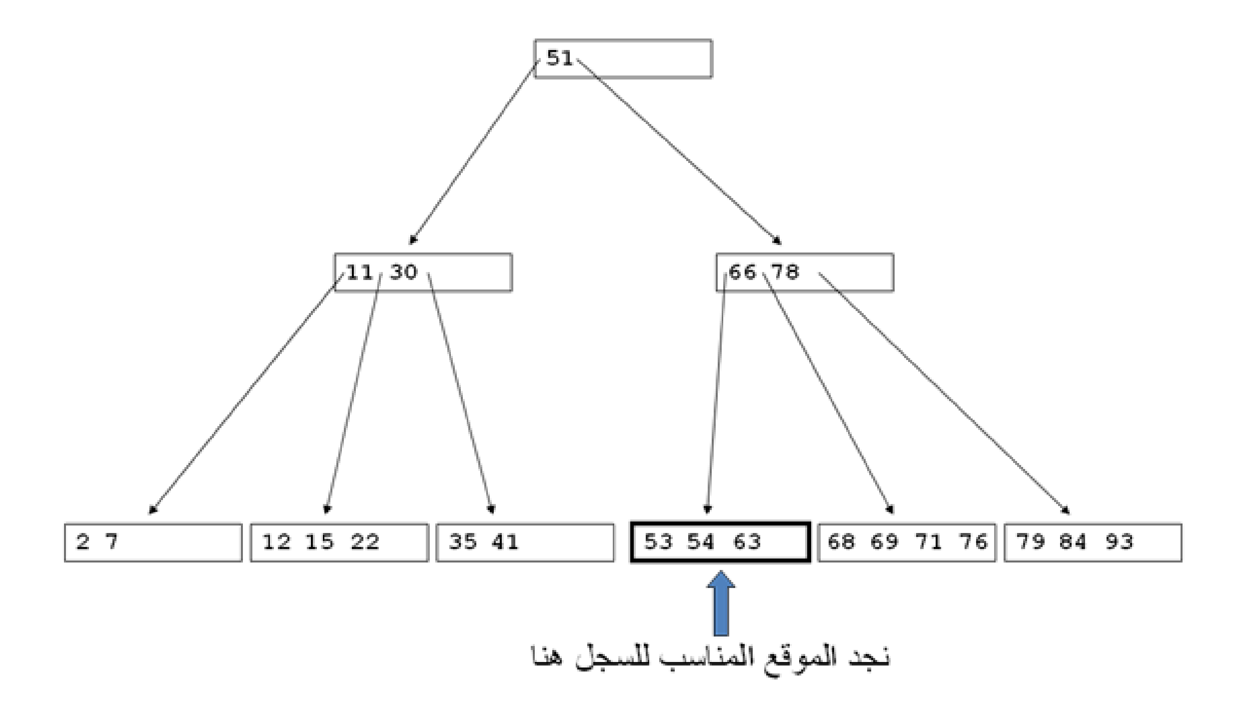
5) Insert <*middle key, ptr to new node*> to the parent.

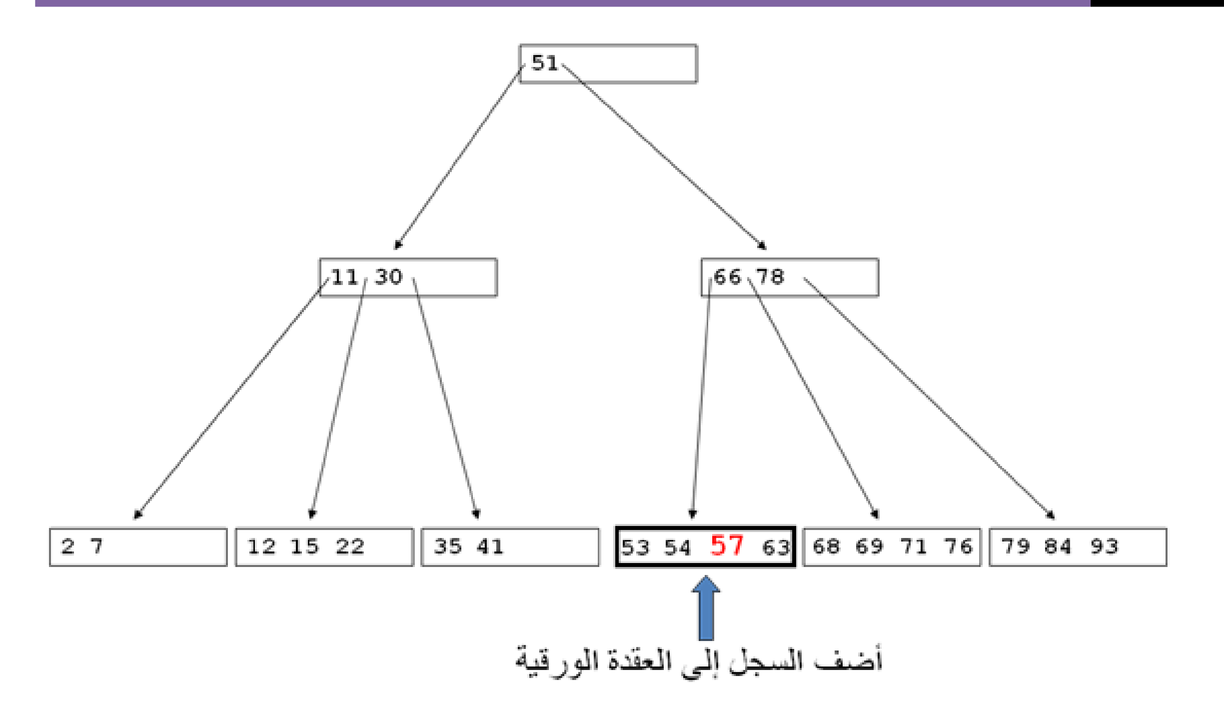
6) Go to Step 3.

مثال:-

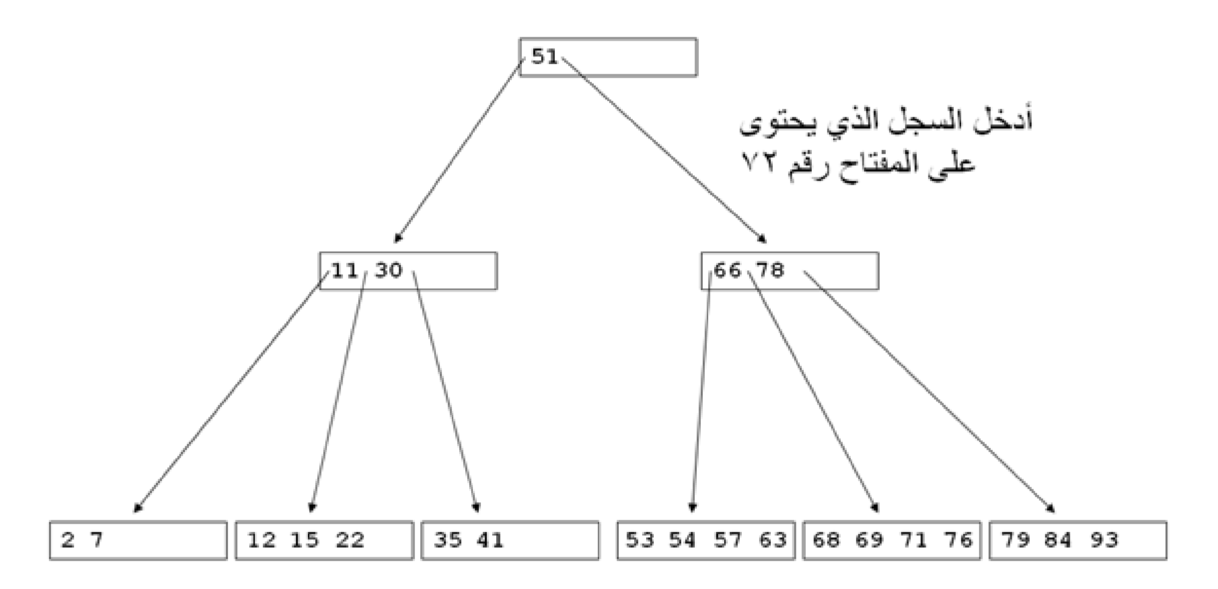


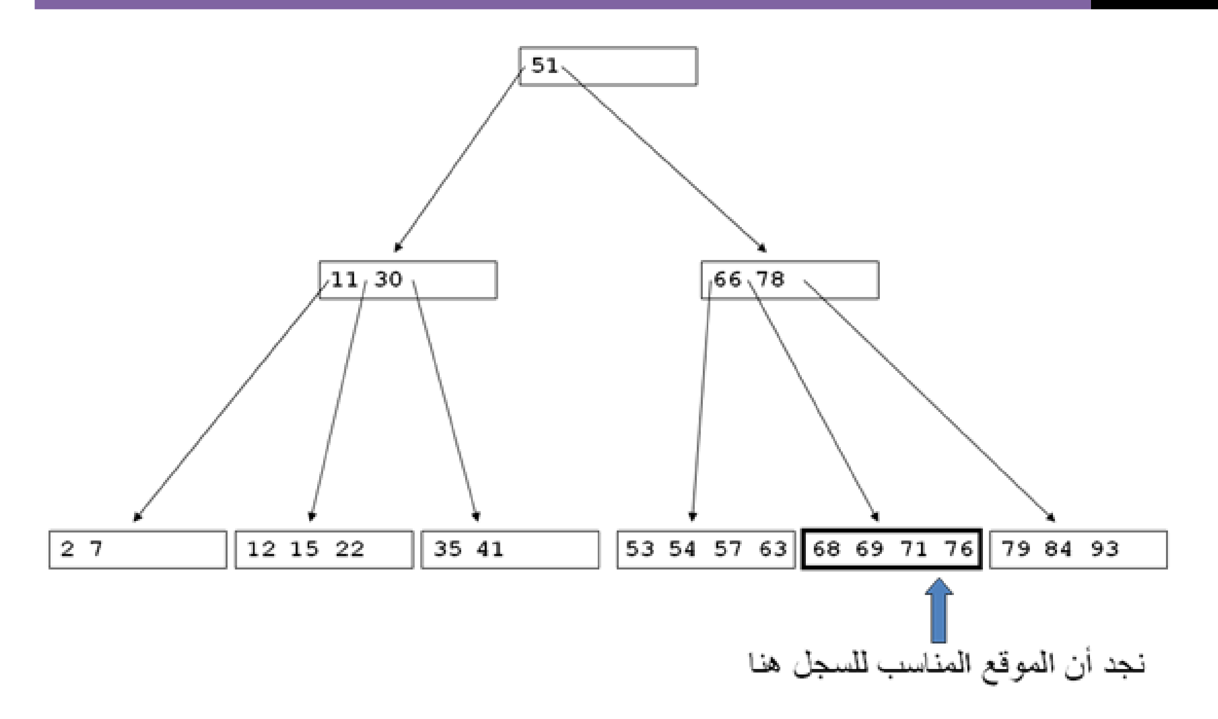
بعد تطبيق الخوارزمية نجد أن :

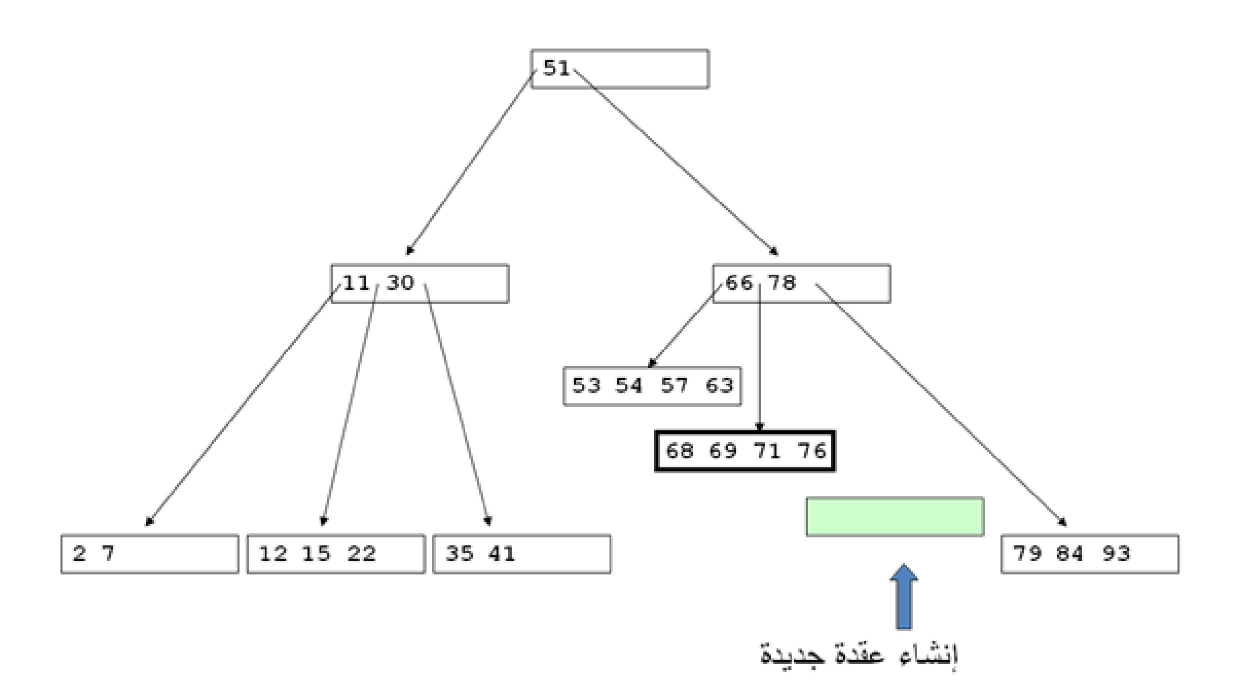


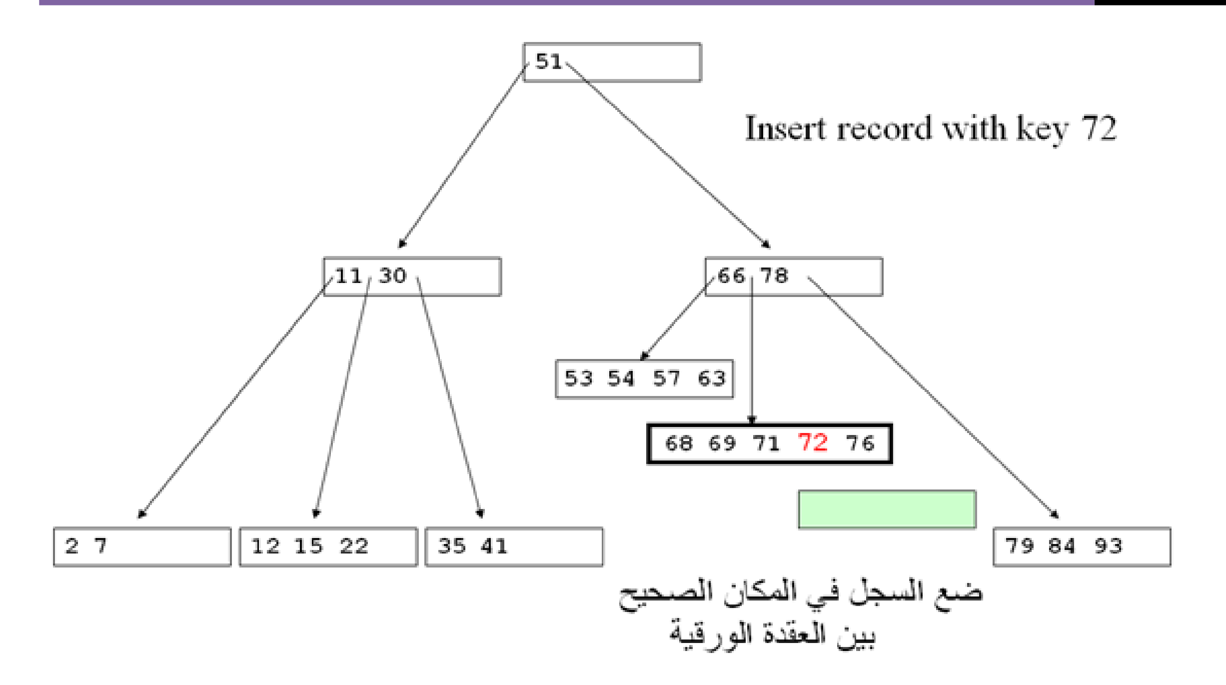

51
11 30
66 78
2 7
12 15 22
35 41
53 54 57 63
68 69 71 76
79 84 93
أضف السجل إلى العقدة الورقية

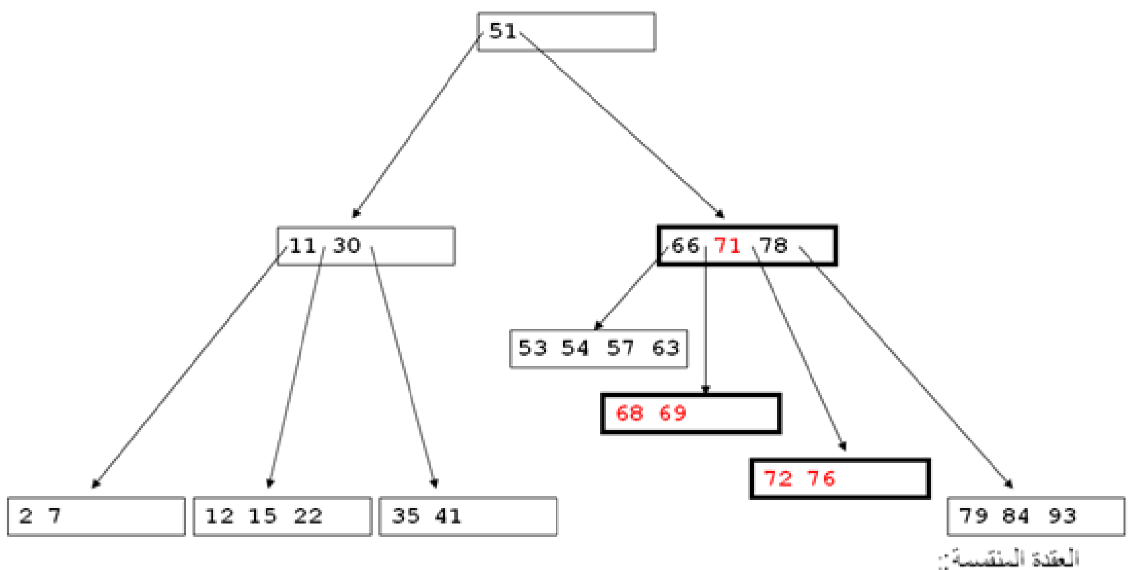
مثال ٢:-


51
أدخل السجل الذي يحتوى
على المفتاح رقم ٧٢
11 30
66 78
2 7
12 15 22
35 41
53 54 57 63
68 69 71 76
79 84 93


51
11 30
66 78
2 7
12 15 22
35 41
53 54 57 63
68 69 71 76
79 84 93
نجد أن الموقع المناسب للسجل هنا


51
11 30
66 78
53 54 57 63
68 69 71 76
2 7
12 15 22
35 41
79 84 93
إنشاء عقدة جديدة



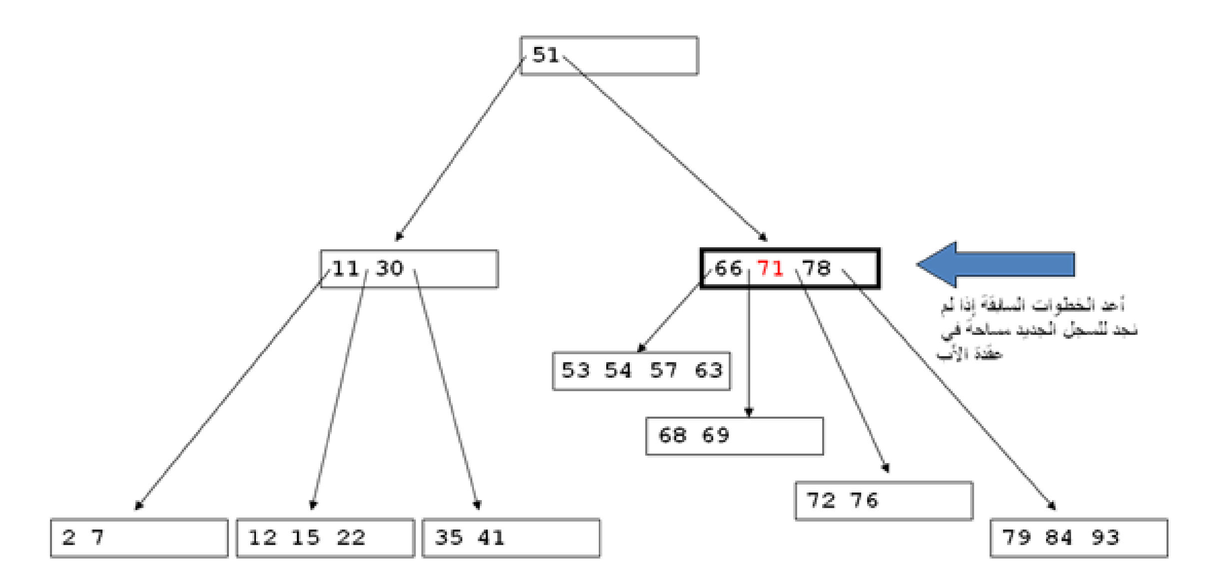


العقدة المنقسمة:

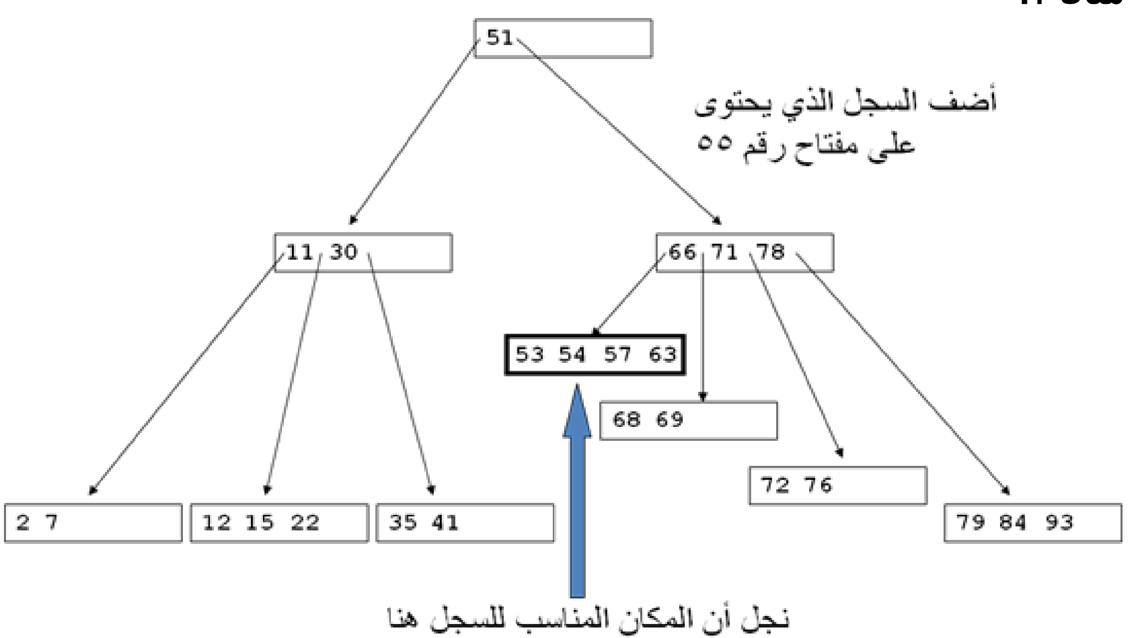
-تبقى السجلات التي أكبر من المفتاح الوسط في العقدة

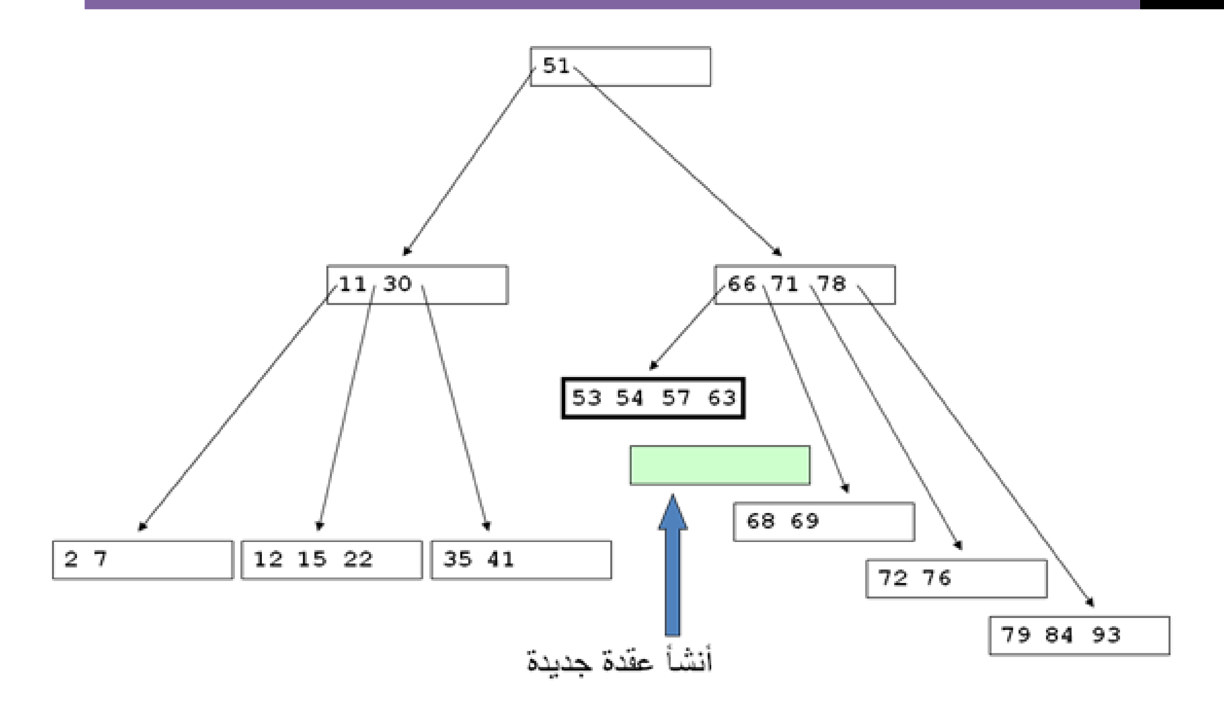
-السجل الأوسط يذهب إلى سجل الأب

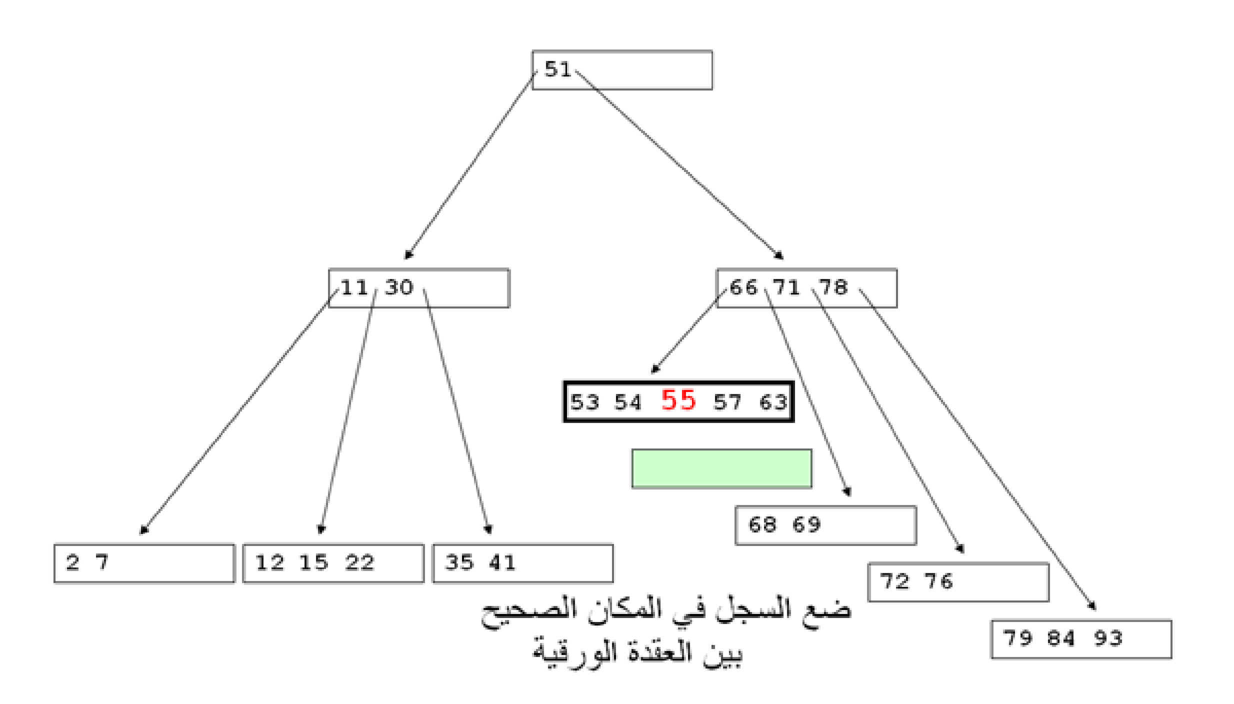
-السجلات الأصغر من سجل الوسط يذهبون إلى العقدة الجديدة

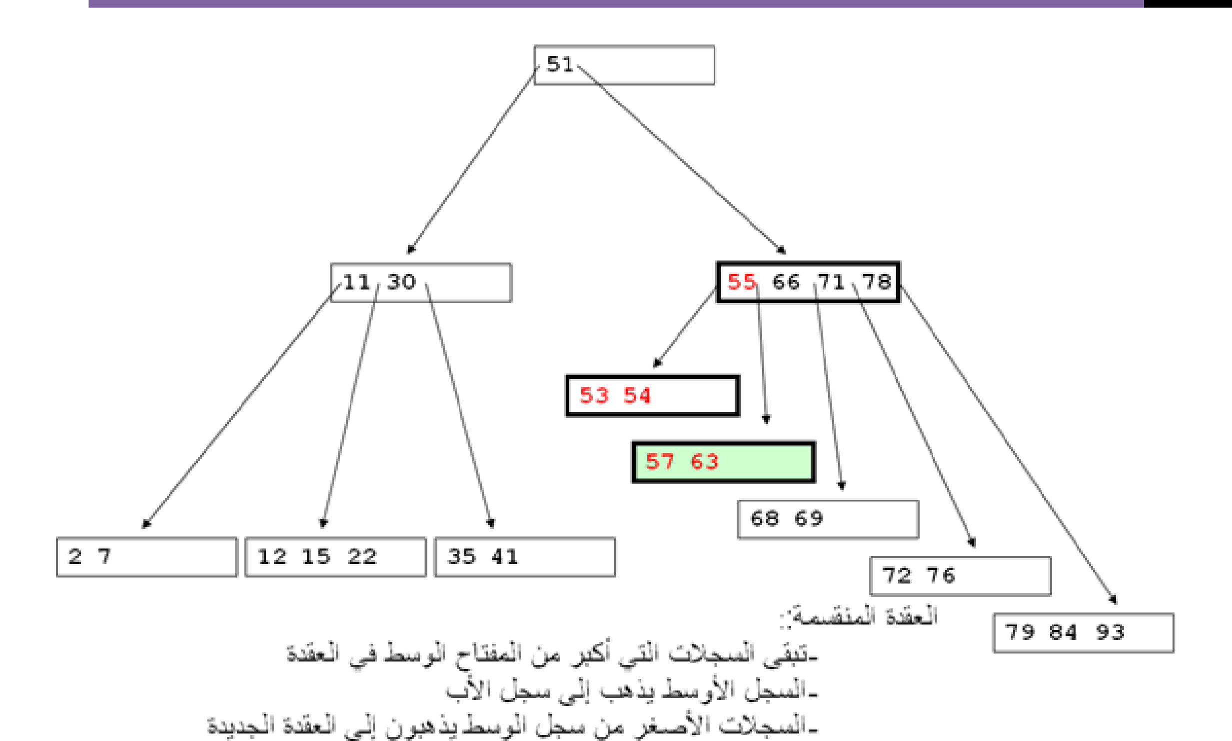

51
11 30
66 71 78
53 54 57 63
68 69
72 76
2 7
12 15 22
35 41
79 84 93
أعد الخطوات السابقة إذا لم
نجد للسجل الجديد مساحة في
عقدة الأب

مثال ٣:-


51
أضف السجل الذي يحتوى
على مفتاح رقم ٥٥
11 30
66 71 78
53 54 57 63
68 69
72 76
2 7
12 15 22
35 41
79 84 93
نجد أن المكان المناسب للسجل هنا


51
11 30
66 71 78
53 54 57 63
68 69
2 7
12 15 22
35 41
72 76
79 84 93
أنشأ عقدة جديدة


51
11 30
66 71 78
53 54 55 57 63
68 69
2 7
12 15 22
35 41
72 76
79 84 93
ضع السجل في المكان الصحيح
بين العقدة الورقية



ثالثاً: عملية المسح :Delete operation

خوارزمية عملية المسح:-

١- أوجد العقدة الورقية الصحية (بإستخدام خوارزمية البحث).

٢- إمسح السجل المراد إزالتة.

٣- إذا كانت العقدة التي تم داخلها عملية المسح نصف إمتلاء و تمت عملية المسح ، إذا كانت غير ذالك إذهب إلى الخطوة التالية:

٤- إذاكان هناك إمكانية الاستعارة سجل من العقدة الشقيق ،تم إذا لم يمكن ذلك إذهب إلى الخطوة التالية:

٥- إدمج العقدة الحالية مع الشقيق

٦- إحذف الفاصل بين العقدة الحالية والعقدة الشقيق و العقدة الأصل.

٧- إذهب إلى الخطوة رقم ٣.

1) Find correct leaf node.

2) Remove the entry from the node.

3) If the node is at least half full, *done*!.

4) Else, possibly *borrow* some entries from a sibling.

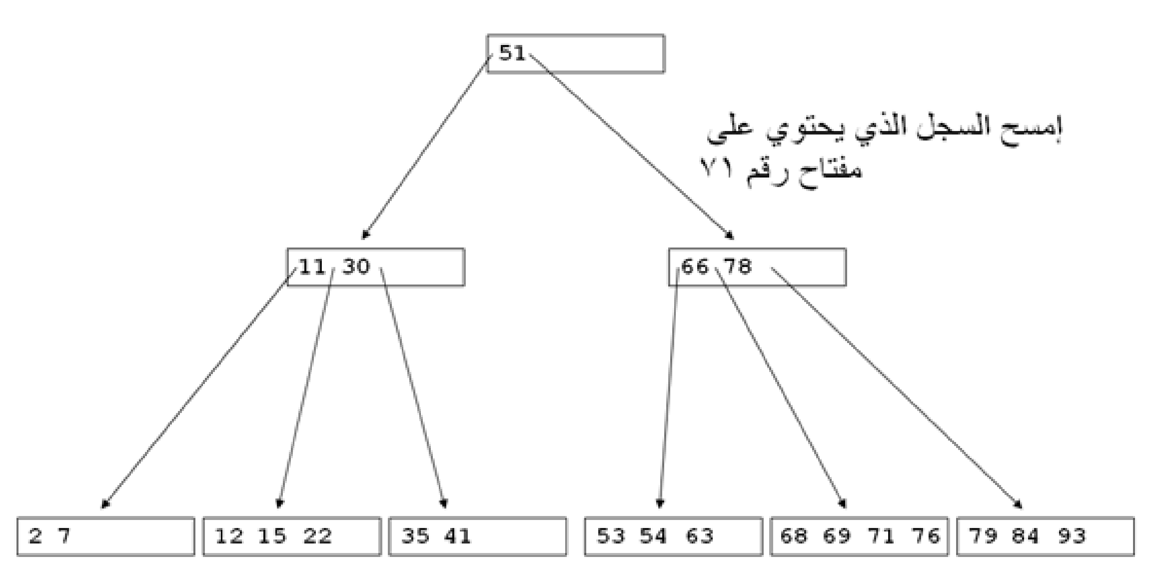
5) If not possible, *merge* the node with the sibling.

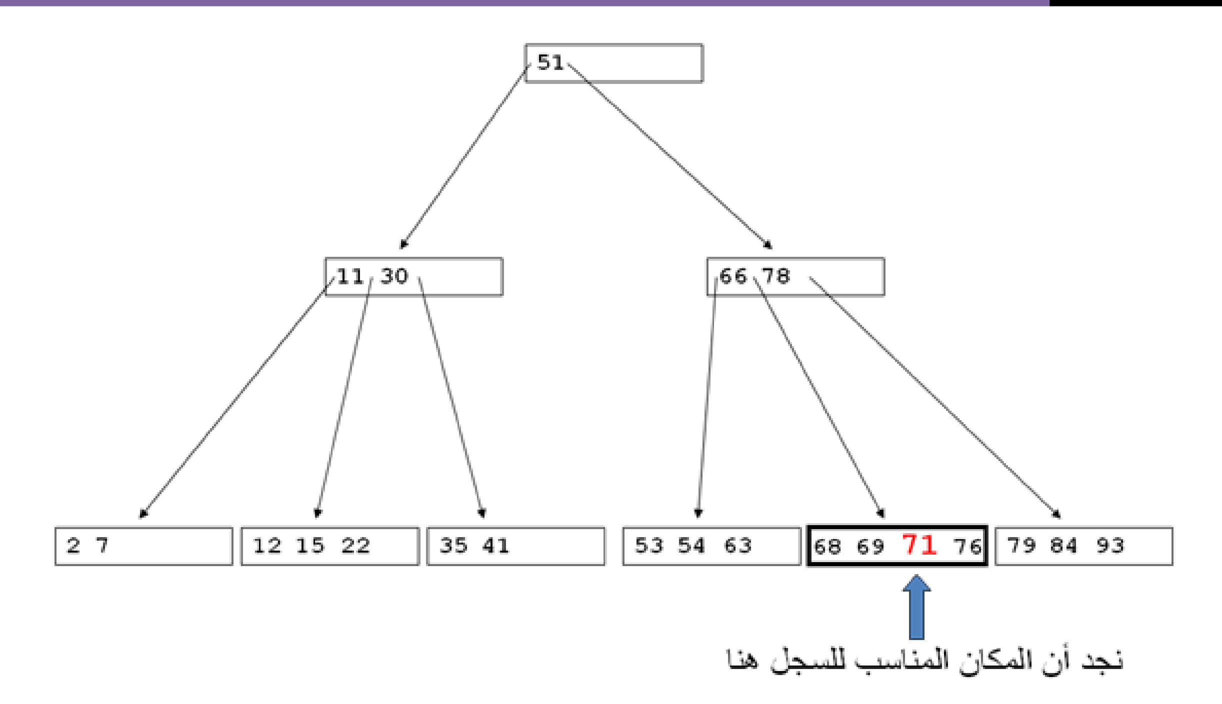
6) Delete the separator between the node and the.

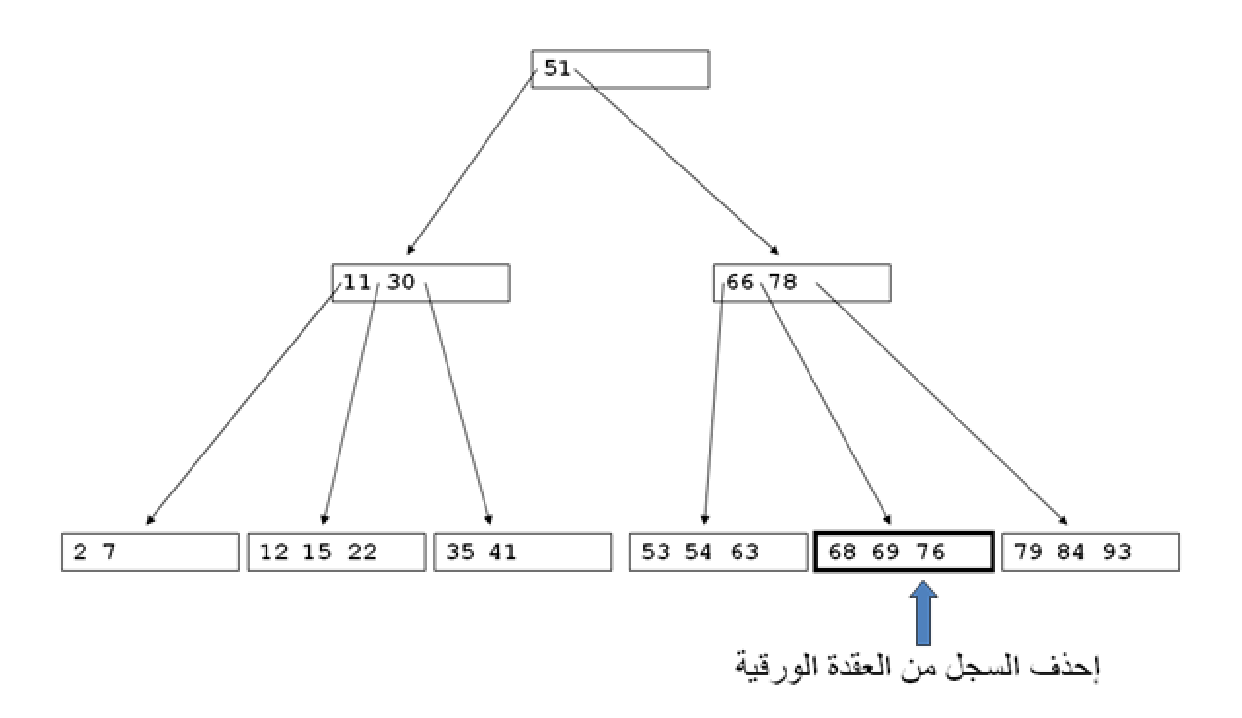
sibling from the parent node.

7) Go to Step 3.

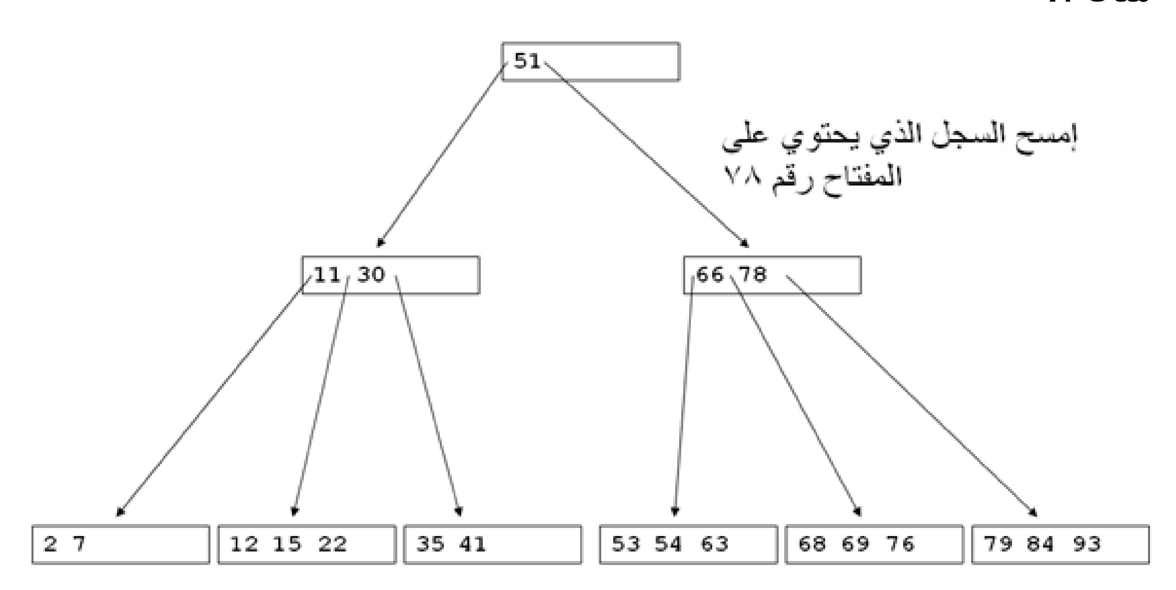
مثال:-



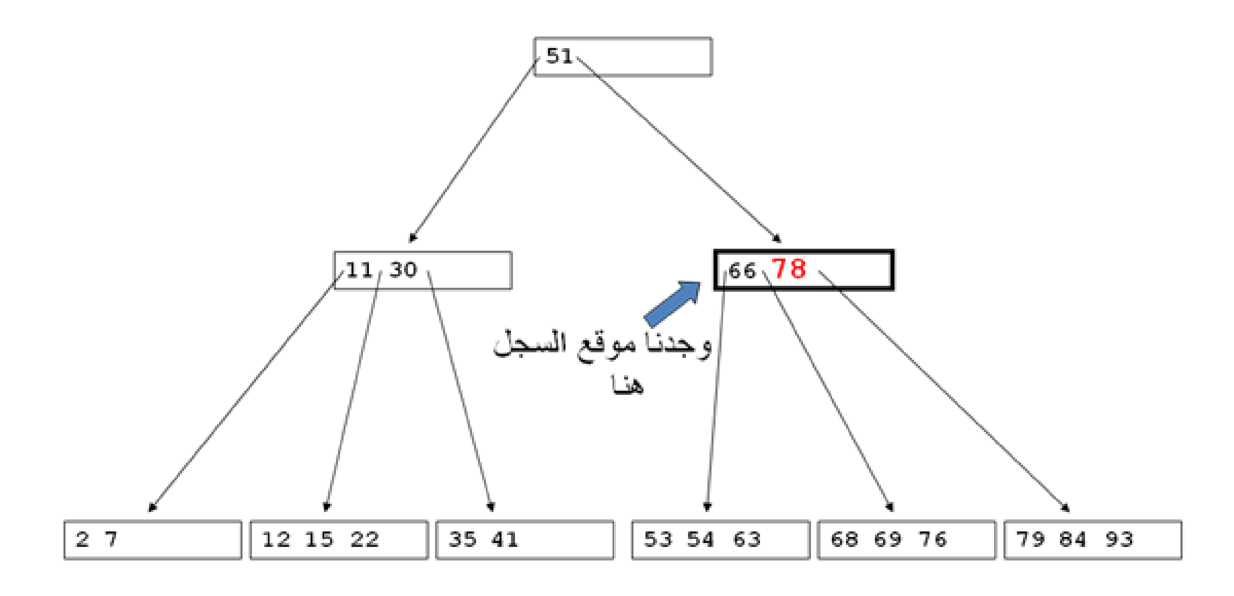

51
11 30
66 78
2 7
12 15 22
35 41
53 54 63
68 69 71 76
79 84 93
نجد أن المكان المناسب للسجل هنا


51
11 30
66 78
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93
إحذف السجل من العقدة الورقية

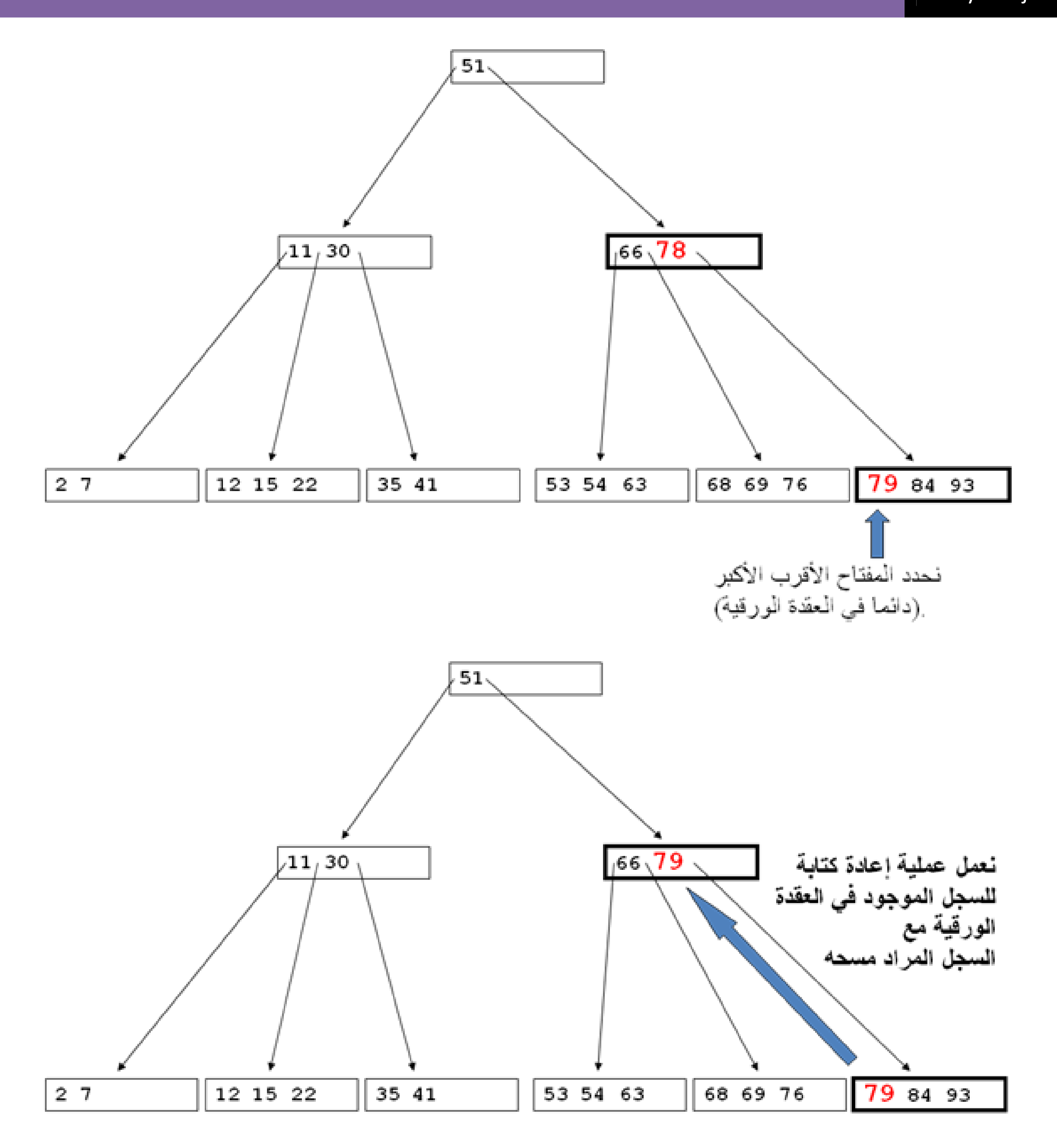
مثال ٢:-

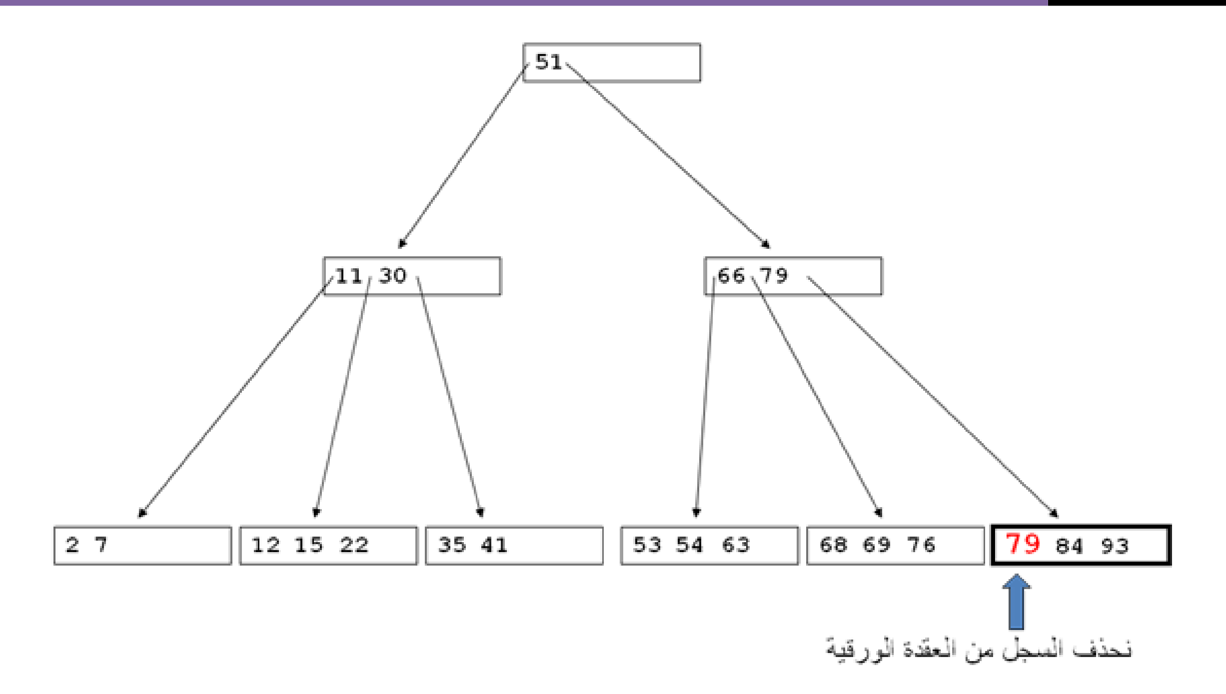


51
11 30
66 78
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93
إمسح السجل الذي يحتوي على
المفتاح رقم ٧٨

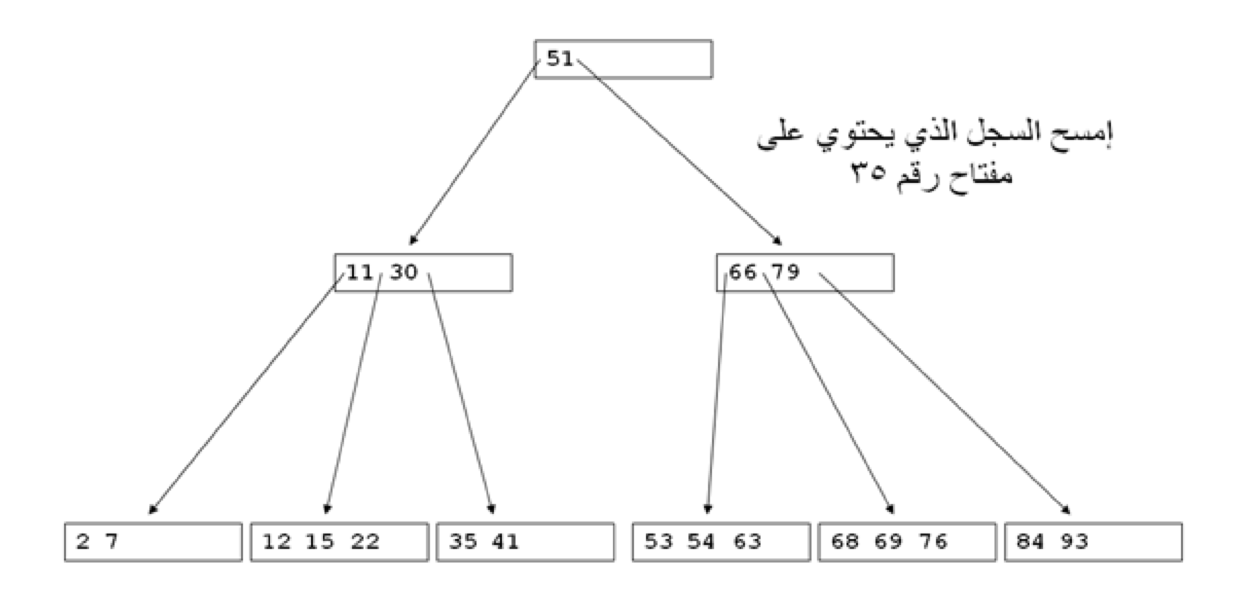


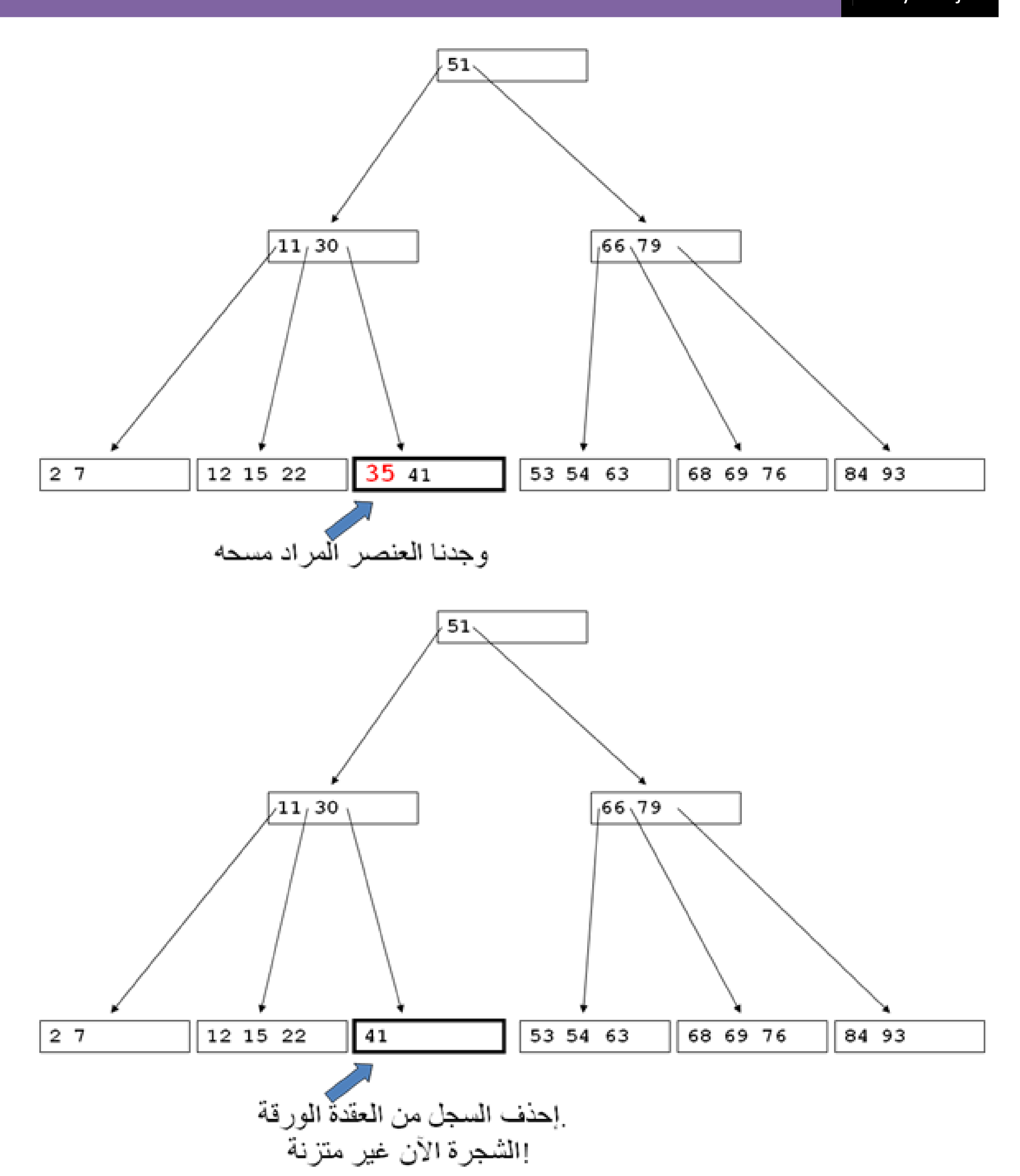
51
11 30
66 78
وجدنا موقع السجل
هنا
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93

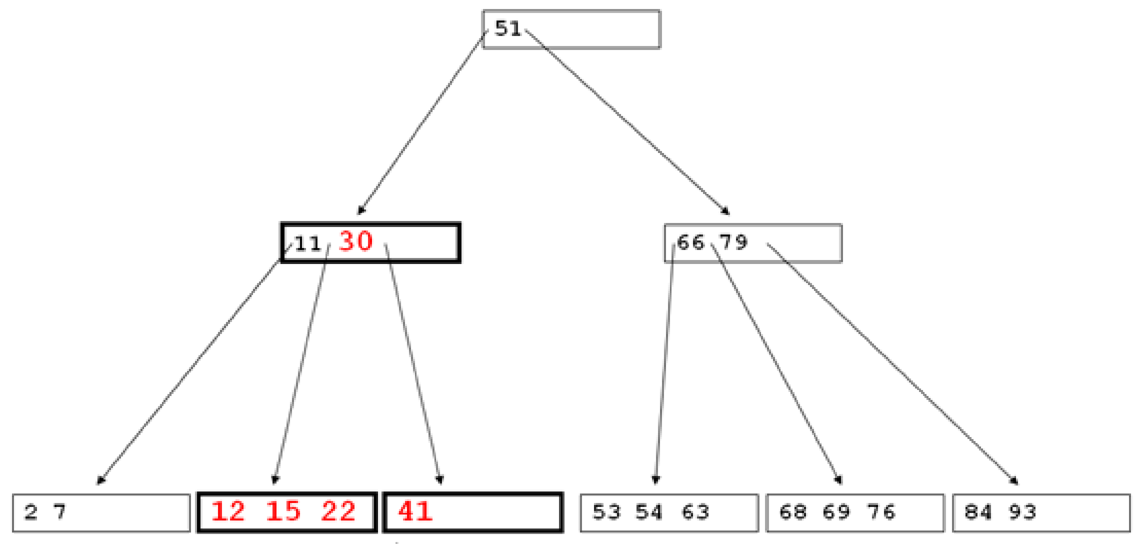

51
11 30
66 78
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93
نحدد المفتاح الأقرب الأكبر
(دائما في العقدة الورقية).
51
11 30
66 79
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93
نعمل عملية إعادة كتابة
للسجل الموجود في العقدة
الورقية مع
السجل المراد مسحه


51
11 30
66 79
2 7
12 15 22
35 41
53 54 63
68 69 76
79 84 93
نحذف السجل من العقدة الورقية

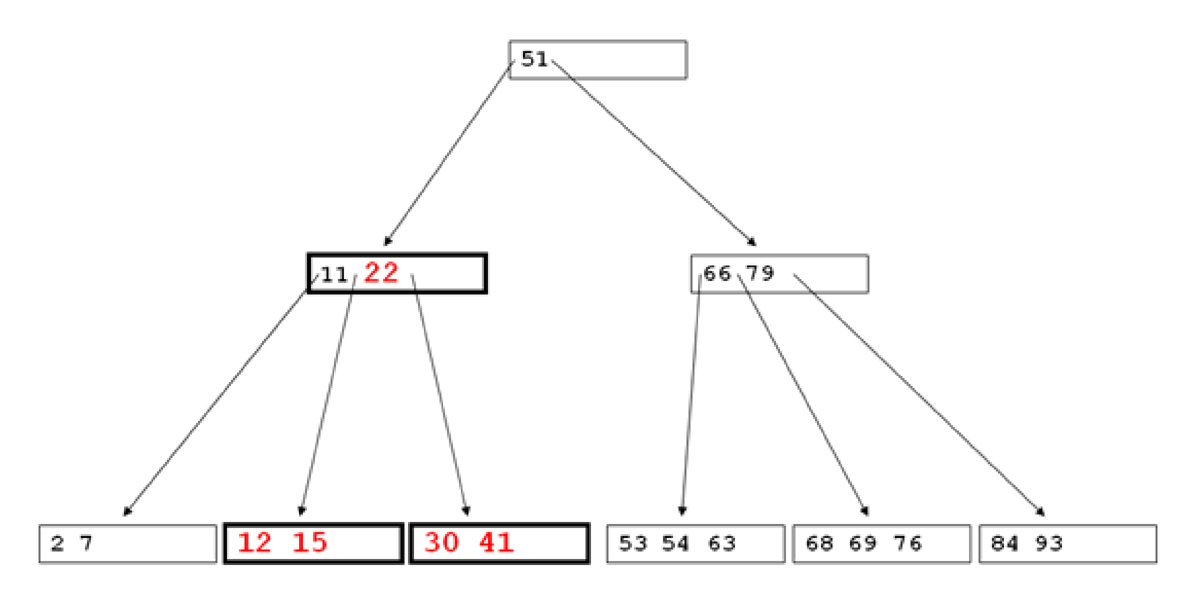
مثال٣:-


51
إمسح السجل الذي يحتوي على
مفتاح رقم ٣٥
11 30
66 79
2 7
12 15 22
35 41
53 54 63
68 69 76
84 93


51
11 30
66 79
2 7
12 15 22
35 41
53 54 63
68 69 76
84 93
وجدنا العنصر المراد مسحه
51
11 30
66 79
2 7
12 15 22
41
53 54 63
68 69 76
84 93
إحذف السجل من العقدة الورقة
الشجرة الآن غير متزنة!

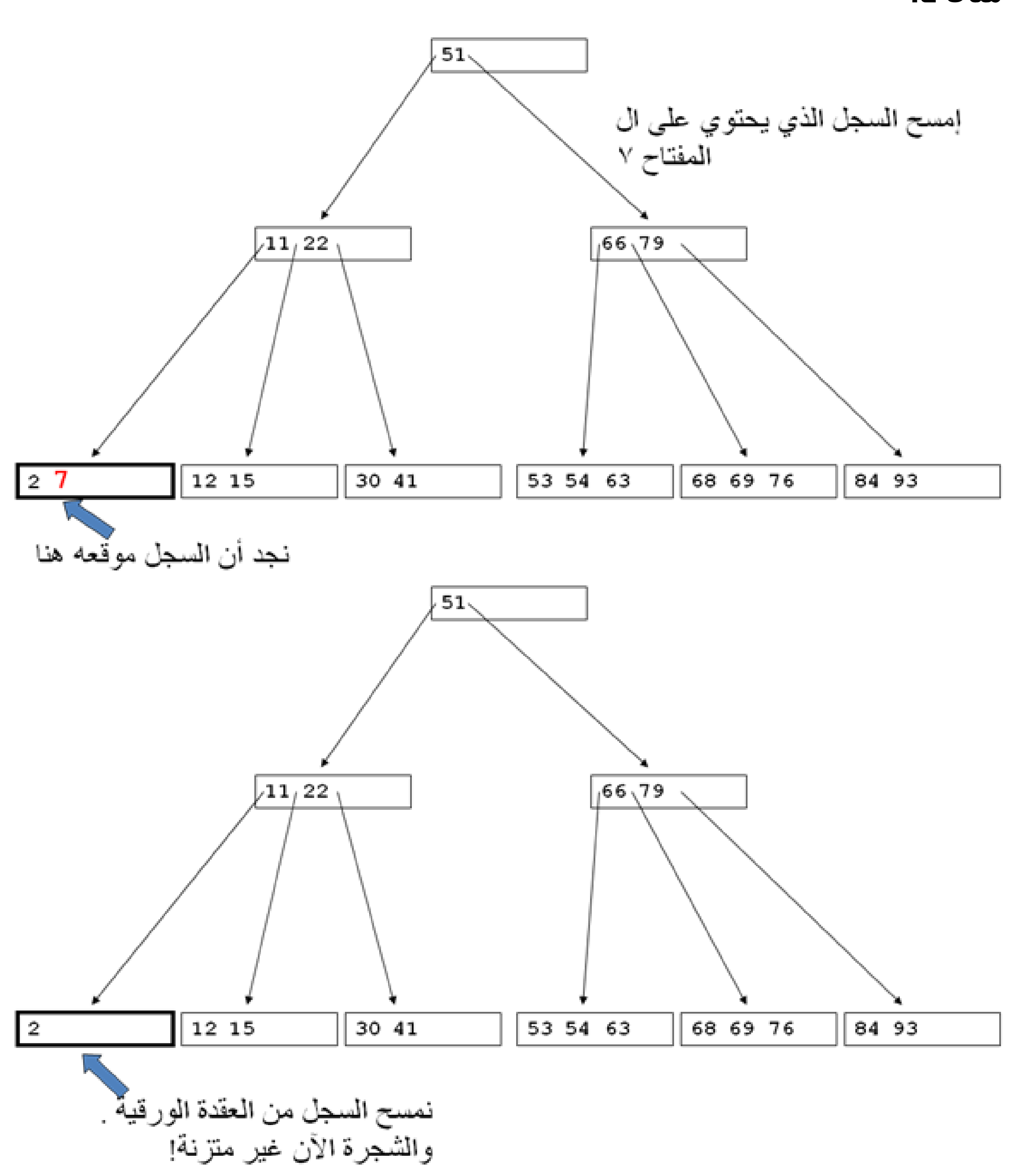


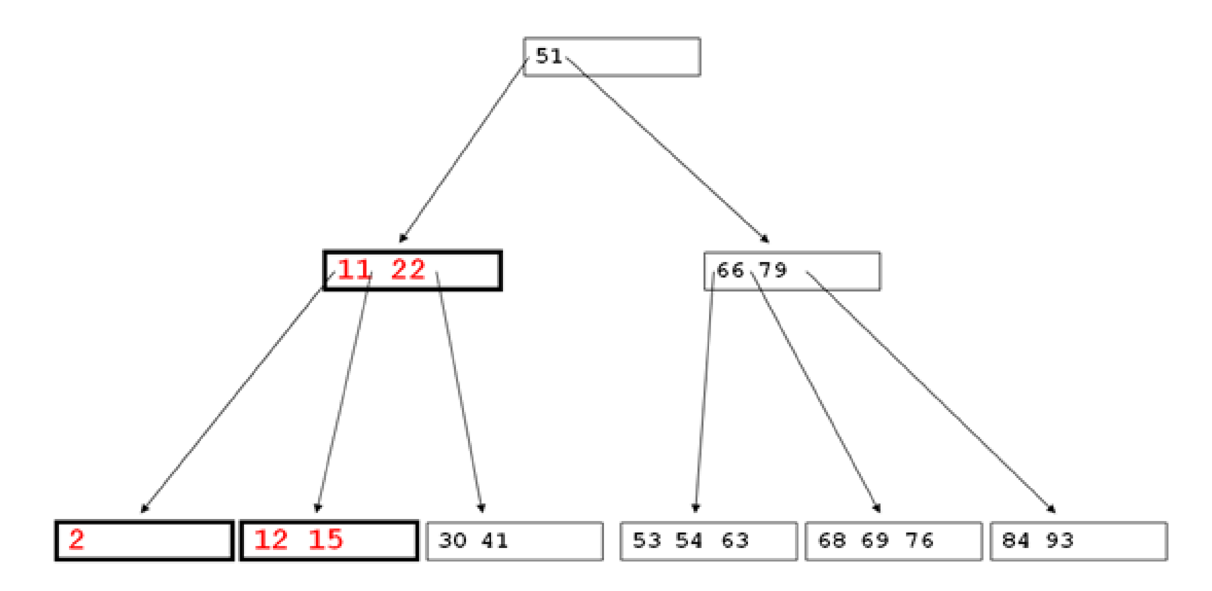
إذا كان للشقيق على الأقل d+1 من السجلات
أعد توزيع السجلات بين :الشقيق
الأصل،و عقدة البحث.



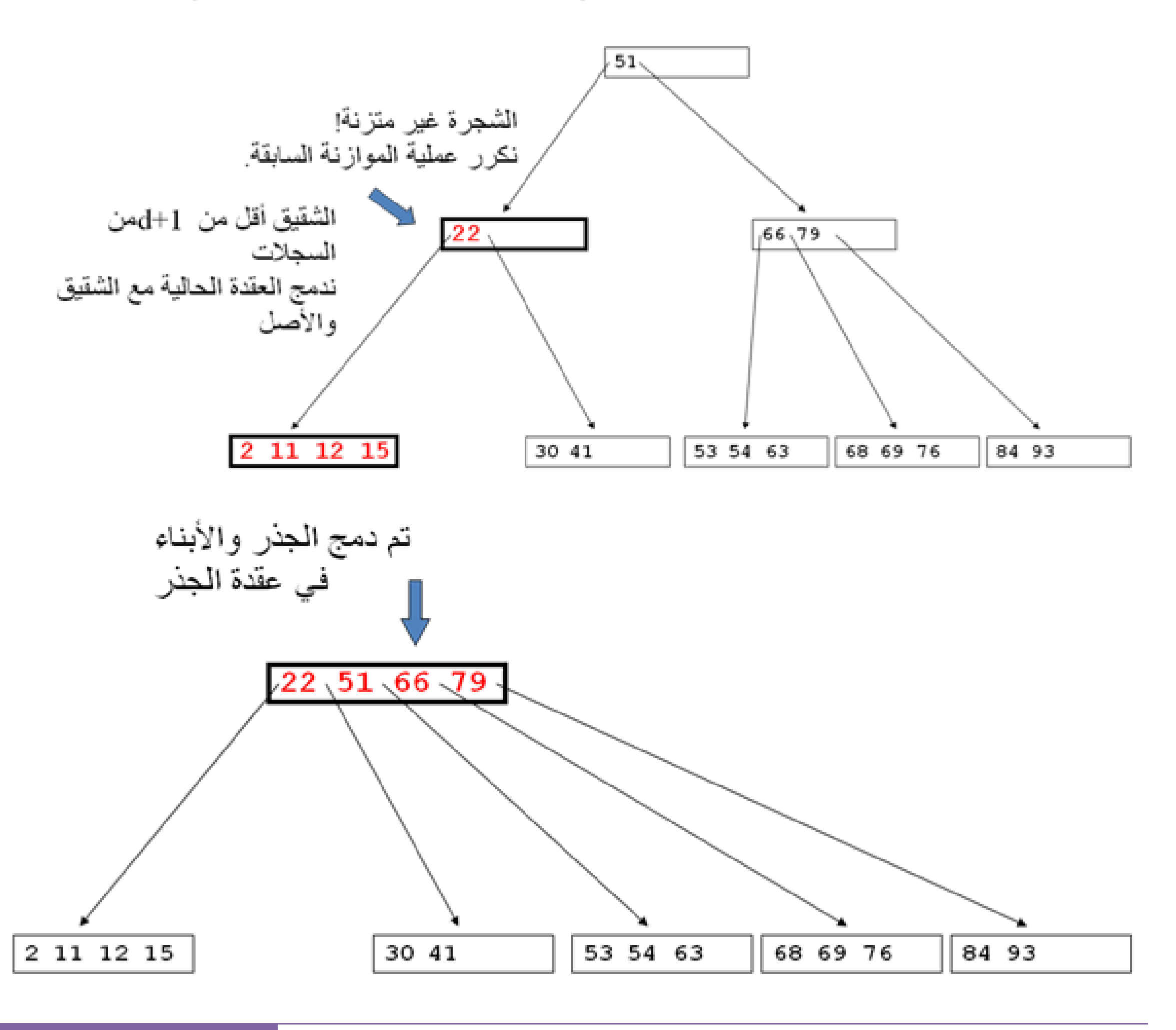
الشجرة الآن متوازية

مثال ٤:-


51
إمسح السجل الذي يحتوي على ال المفتاح ٧
11 22
66 79
2 7
12 15
30 41
53 54 63
68 69 76
84 93
نجد أن السجل موقعه هنا
51
11 22
66 79
2
12 15
30 41
53 54 63
68 69 76
84 93
نمسح السجل من العقدة الورقية .
والشجرة الآن غير متزنة!



الشقيق قليل جدا > d+1سجلات
نقوم بدمج كل من العقدة الحالية و الشقيق مع عقدة الأصل



Program :-

```
program btree2;
const
max=4;
min=2;
type
pointer=^node;
position=0..max;
node=record
     count:position;
     key:array[1..max]of integer;
     branch:array[position] of pointer;
end;


var
yr,root,xr,p,targetnode:pointer;
newkey,newkey2,target,no,n,x,y:integer;
found,pushup:boolean;
targetpos,k:position;



procedure searchnode(target:integer;p:pointer;var found:boolean;
                     var k:position);
begin
with p^ do
  if target<key[1] then
  begin
    found:=false;
    k:=0;
  end

  else  begin
          k:=count;
```

```
          while (target<key[k]) and (k>1) do
            k:=k-1;
          found:=true;
          target:=key[k];
          writeln('yes this key is found, & the key is: ',target)
        end
end;


procedure search(target:integer;root:pointer;var found:boolean;
                 var targetnode:pointer; var targetpos:position);
begin
if root=nil then
  begin
  found:=false;
  writeln('no key in this tree..');
  end

else  begin
        searchnode(target,root,found,targetpos);
        if found then
          begin
            targetnode:=root;
            writeln('this target is found..',target);
          end
        else

search(target,root^.branch[targetpos],found,targetnode,targetpos)
      end
end;

procedure  pushin(x:integer;xr,p:pointer;k:position);
var
i:position;
begin
  with p^ do
```

```
    begin
      for i:=count  downto  k+1  do
        begin
          key[i+1]:=key[i];
          branch[i+1]:=branch[i];
        end;

      key[k+1]:=x;
      branch[k+1]:=xr;
      count:=count+1
    end
end;

procedure split(x:integer;xr,p:pointer;k:position;var y:integer;
                var yr:pointer);
var
i,mediam:position;
begin
if k<=min then
  mediam:=min
else
  mediam:=min+1;
new(yr);
with p^ do
  begin
    for i:=mediam+1 to max do
    begin
      yr^.key[i-mediam]:=key[i];
      yr^.branch[i-mediam]:=branch[i]
    end;
    yr^.count:=max-mediam;
    count:=mediam;
    if k<=min then
      pushin(x,xr,p,k)
    else
      pushin(x,xr,yr,k-mediam);
```

```
    y:=key[count];
    yr^.branch[0]:=branch[count];
    count:=count-1
  end
end;

procedure pushdown(newkey:integer;p:pointer;var pushup:boolean;
                   var x:integer; var xr:pointer);
var
k:position;
found:boolean;
begin
if p=nil then
  begin
    pushup:=true;
    x:=newkey;
    xr:=nil
  end

else  begin
        searchnode(newkey,p,found,k);
        if found then
          {I am change this statment}
          begin
            writeln('Error:inserting dublicate  key');
            write('enter new key to insert: ');
            readln(newkey2);
            pushdown(newkey2{},p^.branch[k],pushup,x,xr)
          end
        else
        if  pushup then
        with p^ do
          if count<max then
          begin
            pushup:=false;
            pushin(x,xr,p,k)
```

```
          end

        else  begin
                pushup:=true;
                split(x,xr,p,k,x,xr);
              end
      end
end;


procedure insert(newkey:integer;var root:pointer);
var
pushup:boolean;
     x:integer;
  xr,p:pointer;
begin
pushdown(newkey,root,pushup,x,xr);
if pushup then
  begin
    new(p);
    with p^ do
      begin
        count:=1;
        key[1]:=x;
        branch[0]:=root;
        branch[1]:=xr;
        root:=p
      end
  end
end;


procedure remove(p:pointer;k:position);
var
i:position;
begin
```

```
with p^ do
  begin
    for i:=k+1 to count do
    begin
      key[i-1]:=key[i];
      branch[i-1]:=branch[i]
    end;
    count:=count-1
  end
end;


procedure successor(p:pointer;k:position);
var
q:pointer;
begin
q:=p^.branch[k];
while q^.branch[0]<>nil do
  q:=q^.branch[0];
p^.key[k]:=q^.key[1]
end;


procedure moveright(p:pointer;k:position);
var
c:position;
begin
with p^.branch[k]^ do
  begin
    for c:=count downto 1 do
    begin
      key[c+1]:=key[c];
      branch[c+1]:=branch[c]
    end;
    branch[1]:=branch[0];
    count:=count+1;
```

```
    key[1]:=p^.key[k]
  end;

with p^.branch[k-1]^ do
  begin
    p^.key[k]:=key[count];
    p^.branch[k]^.branch[0]:=branch[count];
    count:=count-1
  end
end;

procedure moveleft(p:pointer;k:position);
var
c:position;
begin
with p^.branch[k-1]^do
  begin
    count:=count+1;
    key[count]:=p^.key[k];
    branch[count]:=p^.branch[k]^.branch[0]
  end;

with p^.branch[k]^ do
  begin
    p^.key[k]:=key[1];
    branch[0]:=branch[1];
    count:=count-1;
    for c:=1 to count do
    begin
      key[c]:=key[c+1];
      branch[c]:=branch[c+1]
    end
  end
end;

procedure combine(p:pointer;k:position);
```

```
var
c:position;
q:pointer;
begin
q:=p^.branch[k];
with p^.branch[k-1] do
  begin
    count:=count-1;
    key[count]:=p^.key[k];
    branch[count]:=q^.branch[0];
    for c:=1 to q^.count do
    begin
      count:=count+1;
      key[count]:=q^.branch[c]
    end
  end;

with q^ do
  begin
    for c:=k to count-1 do
    begin
      key[c]:=key[c+1];
      branch[c]:=branch[c+1]
    end;
    count:=count-1
  end;
dispose(q)
end;

procedure restore(p:pointer;k:position);
begin
if k=0 then
  if p^.branch[1]^.count>min then
    moveleft(p,1)
  else
    combine(p,1)
```

```
  else  if k=p^.count then
          if p^.branch[k-1]^.count>min then
            moveright(p,k)
          else
            combine(p,k)
  else  if p^.branch[k-1]^.count>min then
          moveright(p,k)
  else  if p^.branch[k+1]^.count>min then
          moveleft(p,k+1)
  else
        combine(p,k)
end;


procedure recdelete(target:integer;p:pointer;var found:boolean);
var
k:position;
begin
if p=nil then
  found:=false
else  with p^ do
      begin
        searchnode(target,p,found,k);
        if found then
          if branch[k-1]=nil then
            remove(p,k)
          else  begin
                  successor(p,k);
                  recdelete(key[k],branch[k],found);
                  if not found then
                    writeln('Target is not found in the cuurrent node');
                end
          else
            recdelete(target,branch[k],found);
          if branch[k]<>nil then
            if branch[k]^.count<min then
```

```
              restore(p,k)
      end
end;


procedure delete(target:integer;var root:pointer);
var
found:boolean;
p:pointer;
begin
recdelete(target,root,found);
if not found then
  writeln('Target is not found')

else  if root^.count=0 then
      begin
        p:=root;
        root:=root^.branch[0];
        dispose(p)
      end
end;


begin {main program}
new(root);
new(targetnode);
{new(p);}
new(xr);
new(yr);
root:=nil;
{root^.count:=0; k:=0;}

writeln('1-->insert');
writeln('2-->search');
writeln('3-->delete');
writeln('4-->exit');
```

```
repeat
writeln('what do you want?');
readln(no);
case no of

1: begin {insert}
writeln('enter newkey to insert in tree:');
readln(newkey);
insert(newkey,root);
end;

2: begin {search}
writeln('enter number of key to search:');
readln(target);
search(target,root,found,targetnode,targetpos);
end;

3: begin {delete}
writeln('enter number of key to delete:');
readln(target);
{delete(target,root);}
end;
end;

until no=4;
if no=4 then
writeln('good by ..');

readln;
end.
```

****

الخاتمه

أ أن تستفيدو من هذا الكتاب لأ تعبت كث ا ا صول على مراجع
باللغة العربية بالنسبة ياكل البيانات لغة باسكال فقررت أن أضع كل الذي تعلمته
أيديكم، وأنصح أي شخص لديه معرفه وبإمكانه نشرها أنه ينشرها لأ ا عباره عن
صدقة جارية له بإذن ا تعا ...

بالنسبة للا امج ا درجة هذا الكتاب سؤاء الأمثلة أو التمارين
تنفيذها كلها والتأكد من صحتها قبل أن أضمنها الكتاب ، وأحببت أن أكثر ال امج
لأن الطرق ا ختلفه للا امج وحـ لل نامج الواحد تعطي إدراك للشخص بالطرق
ا ختلفه للا ه وهي تساعد على الفهم كث ا ، وإذا إستطعنا أن نفكر بشكل صحيح
للا ه فبإذن ا نستطيع بعدهاكتابة أي برنامج مهمها كان معقداً وأقول ذلك لكم عن
ربة وأحببت أن أنقل رب لكم لكي تستفيدو منها ، وإذا كان هناك أي إستفسار
كنكم إرساله إ : sooofy@hotmail.com .

أسأل ا تعا ولكم التوفيق بالدنيا والآخره ،،،،

ولاتنسو من صا الدعوات ولوالدي ، أختكم صفية ناجح .

الفهرس

- **الوحده الثامنه :**

- **الوحده التاسعه :**

- **الوحده العاشره :**